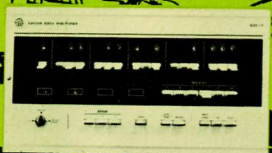
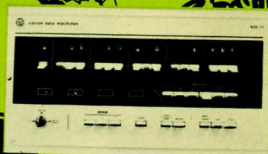
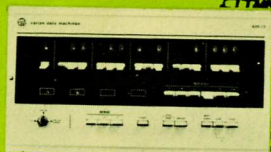


varian 620/f computer handbook



**varian
data machines**

The Big Company in Small Computers

varian 620/f computer handbook



varian data machines
2722 Michelson Drive
Irvine, California 92664

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1	TABLE OF CONTENTS 1-1
2	INTRODUCTION 2-1
	Scope of Handbook 2-1
	Support Documentation 2-1
3	COMPUTER CONCEPTS 3-1
	Computer Capabilities 3-1
	Computer Organization 3-2
	Computer Operation 3-3
	Numbering Systems 3-4
	2's Complement Arithmetic 3-5
	Programming 3-5
	Example Program 3-7
4	VARIAN 620/L GENERAL DESCRIPTION 4-1
5	SYSTEM APPLICATIONS 5-1
6	VARIAN 620/L SPECIFICATIONS 6-1
7	MODEL NUMBERS 7-1
8	CENTRAL PROCESSOR UNIT 8-1
	CPU Section 8-1
	Memory Interface 8-4
	I/O Interface 8-4
9	MEMORY 9-1.
	Master Memory System 9-1
	Slave Memory System 9-1
	CPU/Memory Interface 9-5
	Memory Protect 9-5
	Wrap-Around Addressing 9-5

Section		Page
10	MAINFRAME OPTIONS	10-1
	Special Instruction Set	10-1
	Memory Protection (MP)	10-1
	Real-Time Clock (RTC)	10-1
	Power Failure/Restart (PF/R)	10-2
	Automatic Bootstrap Loader	10-3
11	INPUT/OUTPUT SYSTEM	11-1
	I/O Options	11-1
	I/O System Functions and Organization	11-1
	I/O Bus Structure	11-1
	I/O Modes of Operation	11-3
	Programmed I/O Operations	11-5
	Device Address Codes	11-17
	Interrupt Structure	11-22
	Interrupt-Initiated I/O	11-22
	Cycle-Stealing I/O	11-25
	Priority Structure	11-28
12	INPUT/OUTPUT OPTIONS	12-1
	Priority Interrupt Module (PIM)	12-1
	Buffer Interlace Controller (BIC)	12-8
	Priority Memory Access (PMA)	12-19
13	PERIPHERALS AND I/O INTERFACES	13-1
	Teletypes	13-1
	High-Speed Paper Tape Equipment	13-4
	Seven-Track Magnetic Tape Equipment	13-7
	Nine-Track Magnetic Tape Equipment	13-10
	Disc Memories	13-13
	Drum Memories	13-19
	Line Printer Equipment	13-21
	Oscilloscope Display System	13-24
	Punched Card Equipment	13-27

<u>Section</u>	<u>Page</u>
Digital Plotter	13-31
Buffered I/O Controller	13-34
Relay I/O Module	13-37
Digital I/O Controller	13-40
Analog Input Modules	13-43
Analog Output Modules	13-47
Analog Power Supply Module	13-50
Dataset Controllers	13-52
Automatic Calling Unit Controller	13-59
Data Communications Controllers	13-61
Varian 620/DC Data Communication System	13-63
 14 SYSTEM OPERATION	 14-1
Normal Operation	14-1
Front Panel Controls	14-1
Cold Start	14-5
 15 PHYSICAL CONFIGURATION	 15-1
Mainframe	15-1
Expansion Chassis	15-7
Power Supplies	15-7
Environmental Requirements	15-7
 16 SYSTEM INTERCONNECTIONS	 16-1
Memory Expansion	16-1
I/O Expansion	16-1
Power Supply	16-6
PIM Interconnections	16-8
BIC Interconnections	16-12
PMA Interconnections	16-12
Connector Pin Assignments	16-16
 17 MAINTENANCE	 17-1
Routine Maintenance	17-1
Diagnostic Programs	17-1
Troubleshooting	17-2
Varian 620/f Troubleshooting Procedures	17-4

<u>Section</u>		<u>Page</u>
18	DATA AND INSTRUCTION FORMATS	18-1
	Data Word Formats	18-1
	Instruction Word Formats	18-2
19	ADDRESSING MODES	19-1
	Immediate Addressing	19-1
	Direct Addressing	19-2
	Indirect Addressing	19-2
	Multi-Level Indirect Addressing	19-3
	Indexed with X Register	19-3
	Indexed with B Register	19-3
	Relative Addressing	19-3
20	INSTRUCTION SET	20-1
	Load/Store Instructions	20-1
	Arithmetic Instructions	20-5
	Logic Instructions	20-11
	Jump Instructions	20-15
	Jump-And-Mark Instructions	20-19
	Execution Instructions	20-23
	Control Instructions	20-27
	Shift Instructions	20-29
	Register-Change Instructions	20-33
	I/O Instructions	20-39
21	VARIAN DAS ASSEMBLERS	21-1
	The DAS Character Set	21-1
	Statement Format	21-2
	Assembler Instructions	21-4
	Assembler Directives	21-8
	Relocatability Rules	21-20
	Source Statement Formats	21-21
	Assembler Output Listing	21-21

<u>Section</u>	<u>Page</u>
DAS 4A Operations	21-23
DAS 8A Operations	21-29
DAS MR Operations	21-29
22 BINARY LOAD/DUMP PROGRAM (BLD II)	22-1
Location of BLD II	22-1
Program Options Prior to Loading	22-1
Object Program Loading Options	22-2
Punching Tapes of Memory Contents	22-2
Loading the Binary Load/Dump Program	22-2
Loading the Object Program Tape	22-3
Punching Program Tapes	22-4
23 DEBUGGING PROGRAM (AID II)	23-1
Loading AID II	23-1
Register and Memory Modification	23-1
Handling Paper Tape	23-2
Magnetic Tape Handling	23-2
24 SOURCE PROGRAM EDITOR (EDIT)	24-1
25 MATHEMATICAL PACKAGE	25-1
26 COMPUTER DIAGNOSTICS (MAINTAIN II)	26-1
Test Executive Program	26-1
Communicating with the Test Executive	26-1
Utility Routines	26-2
27 VORTEX	27-1
28 MASTER OPERATING SYSTEM (MOS)	28-1
29 FORTRAN IV	29-1
30 BASIC	30-1
31 REPORT PROGRAM GENERATOR IV (RPG IV)	31-1

<u>Section</u>	<u>Page</u>
APPENDIX	A-1
Standard Character Codes	A-2
TTY Character Codes	A-5
Powers of Two	A-7
Octal-Decimal Integer Conversions	A-8
Octal-Decimal Fraction Conversions	A-12
Alphabetic Index of Instructions	A-15
Numerical Index of Instructions	A-23

SECTION 2 – INTRODUCTION

This is your copy of the Varian 620/f Computer Handbook.

Like the computer it describes, the Handbook is designed for a wide variety of user applications. The Varian 620/f computer has the power and the flexibility to perform the most sophisticated computational tasks. At the same time, the computer has been made as simple as possible to program, operate, and maintain.

Similarly, the material in this Handbook has been arranged so that the experienced programmer or systems engineer can refer directly to those sections that relate to his work, while the less experienced reader will find the book a valuable introduction to computer technology.

Since the Handbook is intended as a useful working tool, any comments or suggestions on the contents would be greatly appreciated. Please direct your remarks to the Publications Department, Varian Data Machines, 2722 Michelson Drive, Irvine, California 92664.

Scope of Handbook

The purpose of the Varian 620/f Computer Handbook is to provide all the basic information necessary for the programming, operation and system interfacing of the Varian 620/f computer.

The Handbook can also serve as a training text for personnel who will be operating systems incorporating the computer.

Maintenance and service personnel, on the other hand, require more detail than is presented in this volume. The information they need is contained in the Varian 620/f Maintenance Manual, available from the Publications Department, Varian Data Machines.

Support Documentation

In addition to the Computer Handbook and the Maintenance Manual, a variety of other documents are available to aid in the programming, operation, and maintenance of Varian 620/f systems.

These include manufacturers' instruction manuals for peripheral devices, such as teletypes and magnetic tape units, separate programming manuals covering such software elements as the FORTRAN IV and BASIC systems, and special manuals for special equipment designed to meet specific user requirements.

So far as possible, the material contained in these separate manuals is not duplicated in this handbook. In the software sections, particularly, only a summary is given, plus a reference to the Varian document providing the complete details.

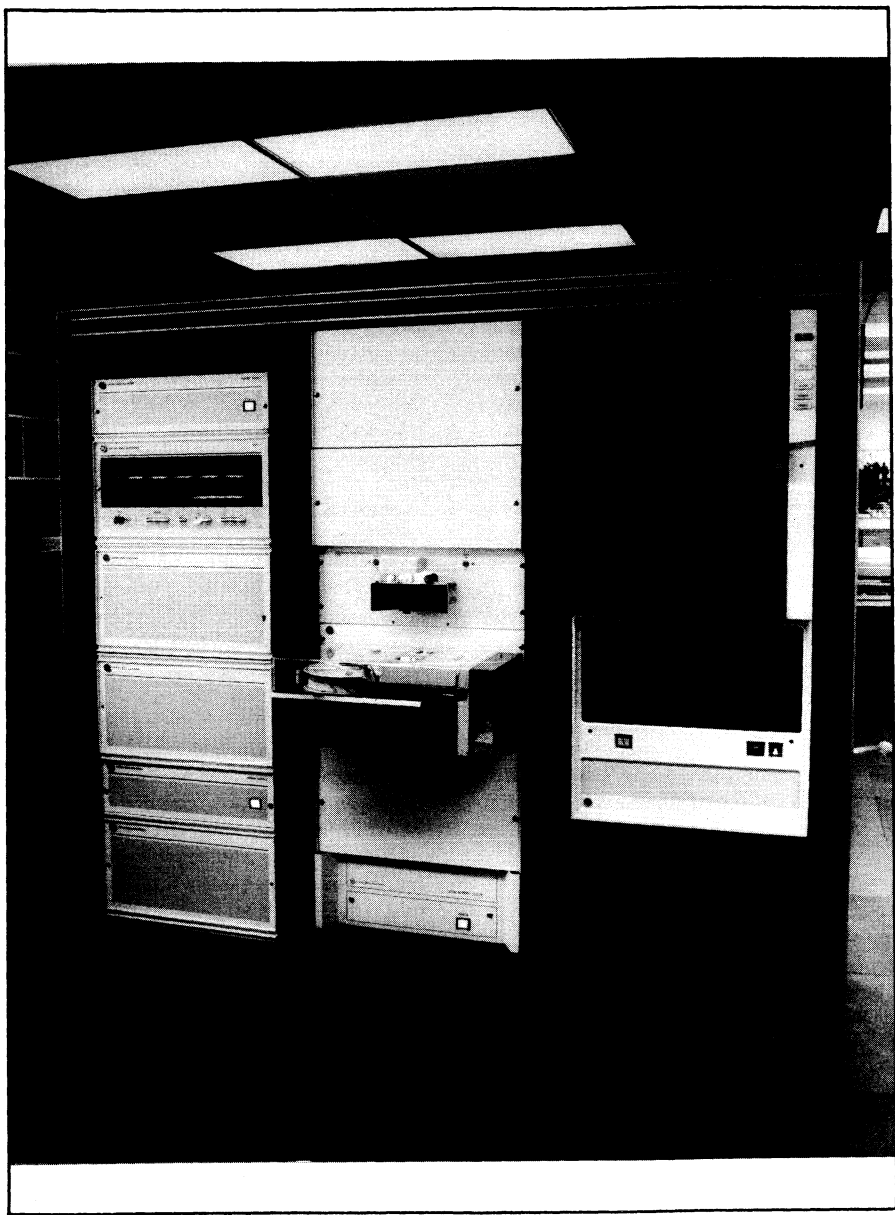


Figure 2-1. Fully Expanded Varian 620/f Computer System

SECTION 3 – COMPUTER CONCEPTS

The purposes of this section are to give the reader a brief overview of the computer concepts on which the design of the Varian 620/f is based, to define terms used in the hardware sections that follow, and to serve as a guide for a further study of these sections.

Computer Capabilities

The Varian 620/f (Figure 2-1) is a general-purpose digital computer. It is general-purpose in the sense that it will perform any type of computational or control task defined by the instructions given to it. In many applications it may serve as a special-purpose, "dedicated" computer, repetitively performing a single, well-defined task such as controlling a chemical process or preparing inventory records. But its general-purpose capabilities are still present and can be utilized at any time.

The computer is "digital" in the sense that it deals entirely in terms of discrete numerical values. Analog values, such as the voltage output of a thermocouple, are converted to digital values before they enter the computer. As a digital computer, the accuracy of the Varian 620/f is absolute, within the limits of its precision.

The precision of a digital computer is established by the number of binary digits (called bits) contained in the "words," or numerical values, processed by the computer. The Varian 620/f is a 16-bit computer. One of these bits, the most significant, is reserved as a sign to indicate whether the value is positive or negative. The largest positive binary number that can be pro-

cessed by the computer is therefore 0 111 111 111 111 111. In decimal terms (see discussion on numbering systems below), this is the equivalent of +32,767, giving the Varian 620/f a precision of one part in 32,767, or approximately $\pm 0.003\%$.

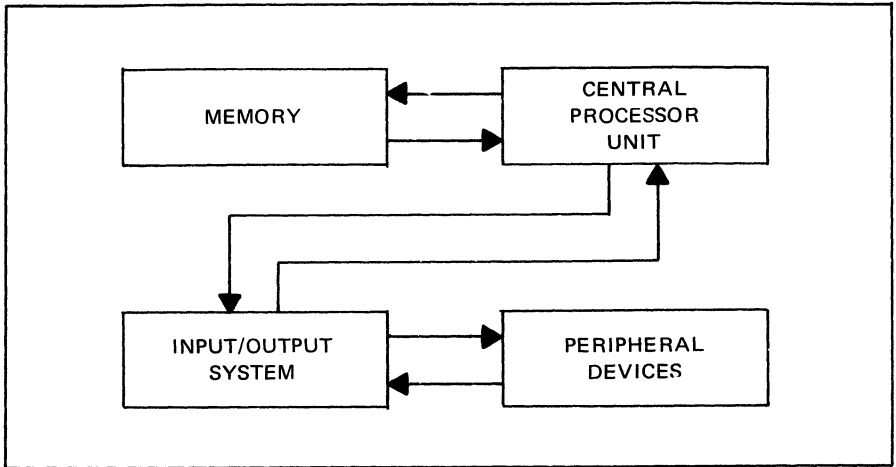
Double-precision arithmetic techniques can be used to process larger numbers when the application requires an even higher degree of precision.

Three other parameters are important in defining the capabilities of a computer. One is the number of discrete instructions that the computer will recognize and respond to. The second is the speed with which the computer performs the operations dictated by the instructions. The third is the size of the memory in which these instructions (and the numerical values that the computer is to process) are stored.

The Varian 620/f can recognize 145 different types of instructions. Many of these contain coding that further define the action to be taken, extending the effective instruction repertoire into the hundreds.

The basic cycle time for execution of an instruction is 0.75 microseconds, or over 1.3 million cycles per second. Individual instructions may require one, two or more cycles to perform.

The Varian 620/f memory is expandable from a minimum of 4096 words to a maximum of 32,768 words. A new memory design and packaging technique allow the memory to be expanded with a high degree of economy, in terms of both dollars and space.



VT11-1227

Figure 3-1. Basic Elements of a Computer System

Computer Organization

Figure 3-1 illustrates the four principal elements that make up a Varian 620/f computer system.

The memory is a storage device that holds 16-bit words, ready for processing by the computer. The words can be either instructions, directing the actions of the computer, or data words to be processed.

The data words can be further divided into two groups: numerical values (or operands) that have been stored in the memory for use during the solution of a specific problem, or constants that remain in memory and are referred to by the computer as required.

In calculating the pay for an individual worker, for example, the hours worked are a variable operand that must be entered into memory, whereas the hourly rate for that type of worker is a constant that is kept in memory.

The central processor unit (CPU) is the

“action” part of the computer. One part of the CPU is the arithmetic unit that performs the additions and subtractions that account for most of the “computations” performed by the computer. Equally important, however, are the CPU’s timing and control functions. The CPU extracts instructions from the memory, acts on them, and transmits the results either back to memory or to the outside world. The CPU controls all the data flow through the computer and provides timing signals that coordinate the actions of every part of the computer system.

The input/output system is the interface or connecting link between the all-electronic computer and the electro-mechanical devices that supply data to the computer or receive the results. The I/O system must coordinate the high-speed responses of the CPU with the comparatively slow actions of a teletype, line printer, disc, or other peripheral device. It must also give the computer the capability of controlling these external devices, starting and stopping motors, or directing that a data word be transmitted or received.

The peripheral devices make up the computer's "outside world," the source and destination of all data processed by the computer. (The operator's control panel is also an interface with the outside world, but its use is normally limited to starting and stopping the computer and in diagnosing any problems that might develop.) A wide range of peripheral devices are available for use in Varian 620/f systems.

The interfacing electronics are all included in the I/O system supplied by Varian. Each peripheral or type of peripheral connects to a peripheral-controller card that plugs into one of the computer chassis. The peripheral is provided with a cable that simply connects to this card.

Each of these four major computer-system elements is described in separate sections of the Handbook. The design and function of the central processor unit is detailed in Section 8. The Varian 620/f memory structure is summarized in Section 9. The I/O system is described in Section 11, and the peripherals and their controllers in Section 13.

Computer Operation

The principal point to remember in understanding the operation of a general-purpose digital computer such as the Varian 620/f is that it accomplishes large, complex tasks by performing a series of small, simple operations. Its power comes in the fact that it performs these operations at a very high rate, running up to hundreds of thousands of separate actions per second.

The computer operation can be compared to the thought processes that occur when a man adds a column of figures in his head. Carrying the analogy to the computer organization described above, the sheet of paper on which the numbers are listed and the

result written is the "peripheral device," the source and destination of the data being processed. The eyes and the hand make up the I/O system. The man's memory serves exactly the same purpose as in the computer, and the portion of the brain that controls his actions and does the actual adding is the CPU.

The numbers enter the memory through the eyes. They are added, one by one, and the result outputted through the hand. But all of this can happen only if the man's memory also contains a program, a set procedure for reading the data, performing the arithmetic operations, and writing the results.

The program was stored in the man's memory at some earlier point in his life when he was taught to add. The program entered his memory by the same I/O channel, his eyes, as the current data. But the program is now fixed in his memory, ready for use whenever he has a column of figures to add.

A computer is programmed in the same way. A set of instructions is assembled that will, when performed in sequence, accomplish a particular task. These are entered into the computer memory through an I/O peripheral device such as a teletype or highspeed tape reader. If the same task is to be used repeatedly, the program will be left in memory for long periods of time. If the computer is used for different purposes on different days, the programs are changed by emptying the first program from memory and replacing it with a different one.

The program instructions are stored in sequential memory locations or "addresses," one instruction word to an address. When the computer is commanded to start by pressing the RUN switch on the control panel, the CPU "fetches" the instruction at the first program address. The bits in

the instruction word are decoded by the CPU to determine the actions that are to be taken. When these have been accomplished, the CPU returns to the memory and fetches the second instruction in the program, decodes it, and performs the indicated actions.

This process continues until every instruction in the program has been executed. The final instruction is usually a command to halt the computer operation. It may also be a command to return to the start of the program and repeat it over again. In such a case, the computer continuously "loops" through the program until it is halted by pressing the STEP switch on the control panel.

Certain actions, such as the series of instructions that will input a character of data from the teletype, are repeated at frequent intervals throughout a program. Rather than use up memory space by repeating the series of instructions each time they are needed, the series is stored once in a separate part of the memory. The program directs the CPU to "branch" to this "subroutine" each time the specified action is needed. The final instructions in the subroutine returns the CPU to the original program.

The instructions recognized by the Varian 620/f CPU are listed and described in Section 20. To simplify their assembly into useful programs, each instruction is given a mnemonic or "name" consisting of three or four letters or digits. These are translated into the binary "machine instructions" that the CPU will recognize by processing them through the Varian 620/f, using a DAS Assembler program (Section 21) supplied by Varian. The symbolic listing prepared by the programmer is called the source program; the machine-code output produced by the Assembler program is called the object program.

The object program is then loaded into the

Varian 620/f, using the Binary Load/Dump program (Section 22) supplied by Varian. Any problems that may develop in the running of the object program may be analyzed by a "debugging" program, AID II (Section 23). Equipment problems can be diagnosed by another program, MAINTAIN II (Section 26). Existing programs can be readily edited and changed by using EDIT (Section 24).

The complete operating sequence for running programs on the Varian 620/f, including the initial "bootstrap" for entering the first instructions into the computer, is described in Section 14.

Numbering Systems

The Varian 620/f internal operations are based on binary (base 2) values. This is true of instructions, operands, and memory address locations. They are all expressed as 16-bit words.

Most problems to be solved by the computer are expressed in decimal digit (base 10) values. Coding to the binary equivalent is normally accomplished as part of the I/O process. The same is true of the decoding back to decimal digits.

Manual binary/decimal conversions may be readily accomplished by simply adding up the decimal values of each "1" in the binary word. The value is determined by its position in the word.

In the case of instruction words, the decimal equivalent of the binary word is not significant since the word is broken into fields for decoding purposes and it is the bit pattern within each field that is important. On the other hand, it would be cumbersome and error-prone if the instructions were always written as 16-bit binary numbers.

Bit Position	Decimal Value
1*	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	256
10	512
11	1024
12	2048
13	4096
14	8192
15	16384
16	32768

*Least significant bit

To resolve this problem (and to eliminate the need to make binary/decimal conversions of memory addresses), Varian 620/f instructions and memory addresses are generally written as octal (base 8) values. The sixteen bits are divided into groups of three, starting from the right or least significant end, and an octal digit used to express each group. The following example shows the technique:

1 010 101 011 001 000 = 125310 (octal)

To reconstruct the binary word, the binary value of each octal digit is listed in series.

If operands or data are expressed in octal, they are normally (as a convention in this Handbook) preceded by a 0 to signify that the number is an octal value. An operand that is not preceded by a 0 is in decimal digits. Octal/decimal conversion tables are given in the Appendix.

2's Complement Arithmetic

The most significant or 16th bit of a Varian 620/f data word is used to indicate whether the value of the word is positive or negative. A "0" in the sign position denotes a positive number; a "1" denotes a negative number.

The negative of a positive number is represented in 2's complement form. The 2's complement of a number may be found in either of two ways:

- Take the 1's-complement of the number (i.e., complement each bit); add "1" in the least-significant bit position.
Example:

+9 0000000000001001

1's-complement 1111111111110110

2's-complement $\begin{array}{r} +1 \\ \hline 1111111111110111 \end{array}$
(-9)

- For an n-bit number (including sign) subtract it from 2^{n+1} . Example:

2^{n+1} 1000000000000000

$\begin{array}{r} -(+9) \\ -9 \\ \hline -0000000000001001 \\ 1111111111110111 \end{array}$

Programming

The preparation of programs for the Varian 620/L is identical, in concept, to the programming of any general-purpose digital computer. The ultimate objective is to prepare a list of machine instructions that will perform the task assigned. The quality of the program can be judged on the basis of the number of instructions required (as few as possible), and the speed with which the computer can complete the task.

The steps to be taken in the writing of an effective, efficient program are as follows:

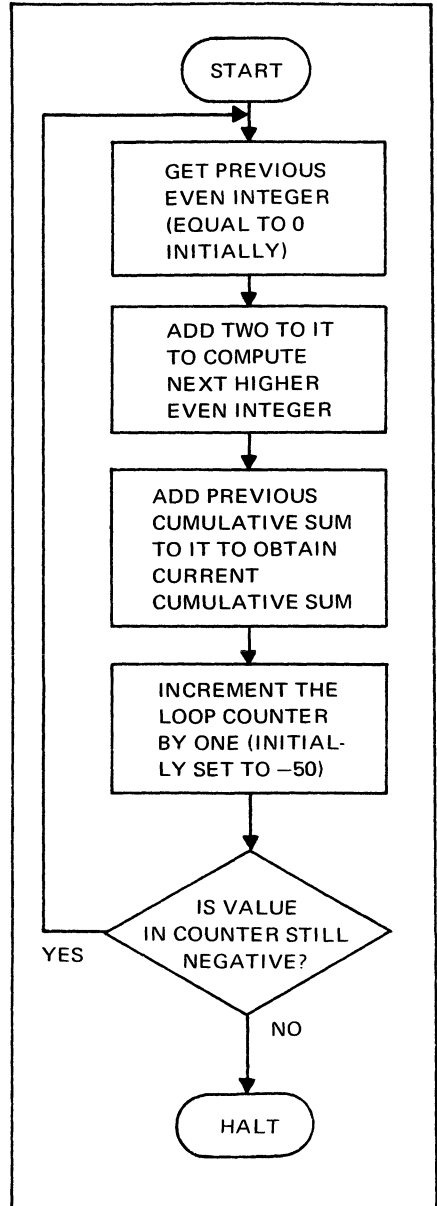
- a. Analyze the problem in terms of the functions that must be performed by the computer, taking into account the capabilities of the Varian 620/f organization and instruction list.
- b. Prepare a flow chart that outlines the logical steps required in solving the problem.
- c. Use the flow chart to prepare a source program in symbolic notations suitable for translation by the Varian 620/f DAS Assembler (see Section 21) or by using FORTRAN IV, BASIC or RPG IV language statements.
- d. Obtain an object program by running the source program with the appropriate assembler or compiler.
- e. Test the program and debug it with the AID II diagnostic program (Section 23).

Example Program

An example of a brief Varian 620/f program is given in Figures 3-2 and 3-3.

The problem to be solved is the summation of the even integers from 2 to 200. The solution is first defined by the operations and logical decision shown in the flow chart illustrated in Figure 3-2. The completed program, shown in Figure 3-3, occupies memory location 004000 through 004017.

The loop counter has an initial value of -50. After each iteration of the program, the counter is incremented by one and tested for negative contents. After the 50th iteration, the counter will change to zero, which is a non-negative quantity, and the program will halt.



VT11-1228

Figure 3-2. Sample Flow Chart

Memory Location	Instruction Code	Label	Instruction Mnemonic	Variable Field	Comment
		*			PROGRAM TO SUM THE
		*			EVEN INTEGERS
		*			FROM TWO TO ONE-
		*			HUNDRED (INCLUSIVE)
004000	014014	LOOP	LDA	NEXT	COMPUTE 1ST (NEXT)
004001	124012		ADD	TWO	HIGHER
004002	054012		STA	NEXT	EVEN INTEGER
004003	124012		ADD	TOTL	ADD IT TO
004004	054011		STA	TOTL	CUMULATIVE TOTAL
004005	014011		LDA	CNTR	DECREMENT THE LOOP
					COUNTER
004006	124004		ADD	ONE	BY ONE AND
004007	054007		STA	CNTR	IF SUM IS NOT COMPLETE
004010	001004		JAN	LOOP	REPEAT
004011	004000 R				
004012	000000		HLT		WHEN DONE, HALT
004013	000001	ONE	DATA	1	AMOUNT TO DECRE-
					MENT LOOP COUNTER
004014	000002	TWO	DATA	2	CONSTANT USED TO
					COMPUTE NEXT EVEN
					INTEGER
004015	000000	NEXT	DATA	0	NEXT EVEN INTEGER
					VALUE
004016	000000	TOTL	DATA	0	CUMULATIVE TOTAL
004017	177716	CNTR	DATA	-50	NEG. CONST. FOR NUM-
					BER OF LOOP
					ITERATIONS
	000000		END		

Figure 3-3. Sample Program

SECTION 4 – GENERAL DESCRIPTION

The Varian 620/f, latest in the evolving line of Varian 620 computers, is the world's fastest, most versatile 16-bit minicomputer. It is also the first all-new computer to be introduced with a full complement of peripherals and supporting software.

750 Nanoseconds Cycle Time

The basic cycle time of the Varian 620/f is 750 nanoseconds, making it the fastest small computer available.

The unprecedented processing speed of the Varian 620/f is the product of a new memory design and imaginative use of high-speed TTL and MSI integrated circuits.

Expanded Instruction Set

A number of powerful new instructions have been added to the Varian 620/f repertoire. The affect of these instructions is to allow the computer to duplicate, through the execution of a single instruction, data processing steps that previously required two, three, or even more program steps. Complementing these instructions are such new standard addressing modes as pre- and post-indexing, extended addressing, and index/indirect addressing.

The combination of increased speed and additional instructions results in a throughput capability up to three times that of earlier Varian 620 models.

Expandable Memory

The basic memory for the Varian 620/f is a 3-wire/3D planar core memory.

The total computer memory can be expanded to a maximum of 32,768 words.

Three Input/Output Techniques

The Varian 620/f features three different types of I/O operations.

The basic I/O bus is a party line that can transfer data at rates up to 30,000 characters per second. A Direct Memory Access mode interlaces program steps with I/O transfers to and from memory to produce data rates up to 276,000 words per second. A direct port to memory is provided by the Priority Memory Access (PMA) for high-speed transfer rates up to 1.3-million words per second.

Full Complement of Peripherals

A Varian 620/f system can include a variety of "off-the-shelf" peripheral devices. Controllers and devices, tested and proved on earlier Varian 620 models, are immediately available for all standard input and output devices, including teletypes, paper-tape equipment, magnetic tape transports, disc packs, line-printers, card-readers, communication controllers, analog/digital front ends, displays, plotters, drums, and special interfaces such as an IBM 360/620 Interface Control Unit.

Flexible Packaging

New packaging techniques provide for economical expansion of the Varian 620/f computer system. A master/slave configuration reduces the cost of memory expansion. I/O controller boards can be added by simply

general description

inserting them into the I/O bus slots provided.

Software Library

All programs for earlier versions of the Varian 620 computer can be operated on the Varian 620/f. The standard library includes a full set of software and programming aids, such as DAS Assemblers, AID II, EDIT, Program Loaders, FORTRAN IV, MAIN-TAIN II, math library, and a Master Operating System (MOS), plus such new pro-

grams as RPG IV, a business-oriented language, and BASIC.

Console Enhancement

Operation of the computer is simplified and made more flexible by such new front-panel console features as Automatic Bootstrap and Operator Interrupt switches and the ability to use the Register Entry switches as 16 additional sense switches. Both power and the console are secured by a single key-operated switch.

SECTION 5 — SYSTEM APPLICATIONS

Applications for the Varian 620/f cover the complete computer spectrum: instrumentation systems, data acquisition systems, time-share networks, and scientific and commercial computing.

A minimum system, suitable for scientific computations within a single laboratory, for example, would consist of the computer mainframe and a single teletype, as shown in Figure 5-1.

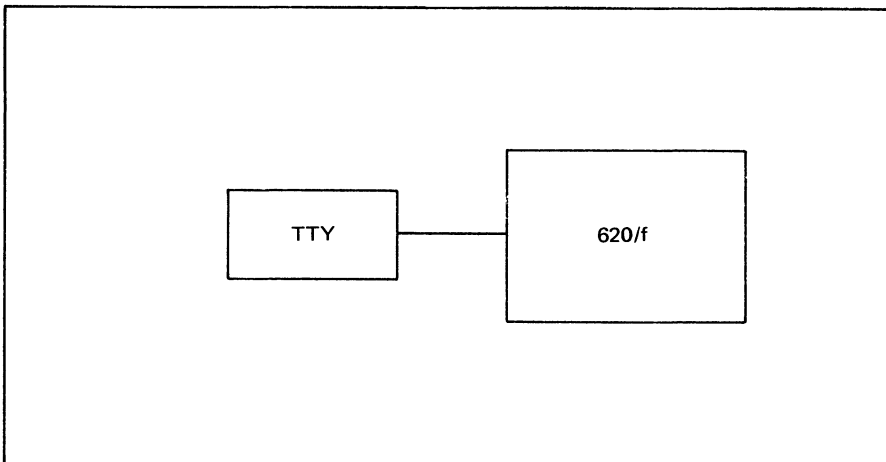
The addition of a high-speed paper-tape reader and punch (Figure 5-2) would extend the utility of the computer to other users, who could rapidly load the computer with their own programs and record the results in a form suitable for later analysis.

In certain cases, the output of a scientific computation is more readily grasped if it is

presented in graphic form. Standard Varian 620/f peripherals include digital plotters and oscilloscope displays, and these could be used (Figure 5-3) in such applications.

Business applications, on the other hand, have a need for a large, efficiently stored, yet readily accessible data base, and a capability for high-speed print-out of checks, invoices, inventory records and similar documents. The system shown in Figure 5-4 meets such needs, using the standard Varian 620/f disc drives or magnetic tape transports as the method for bulk data storage and a line printer for the output.

A variation on this type of application is shown in Figure 5-5. This is the basic configuration for the Varian Report Program Generator (RPG IV) system described in Section 31.



VT11-1229

Figure 5-1. Minimum System for Scientific Applications

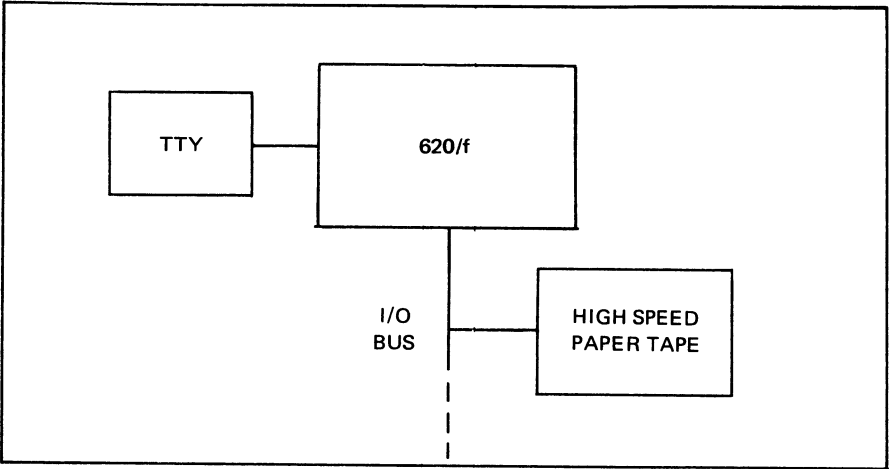
system applications

The speed and flexibility of the Varian 620/f also make it applicable to real-time process monitoring and control functions. Figure 5-6 illustrates a real-time process control system for a chemical plant or petroleum refinery. Figure 5-7 is an equivalent system for a manufacturing plant.

The use of the Varian 620/f as a data-communication switching center is shown in Figure 5-8. The Varian 620/DC Data Communication System described in Section 13 is a highly efficient and versatile unit, capable of interfacing up to 128 communi-

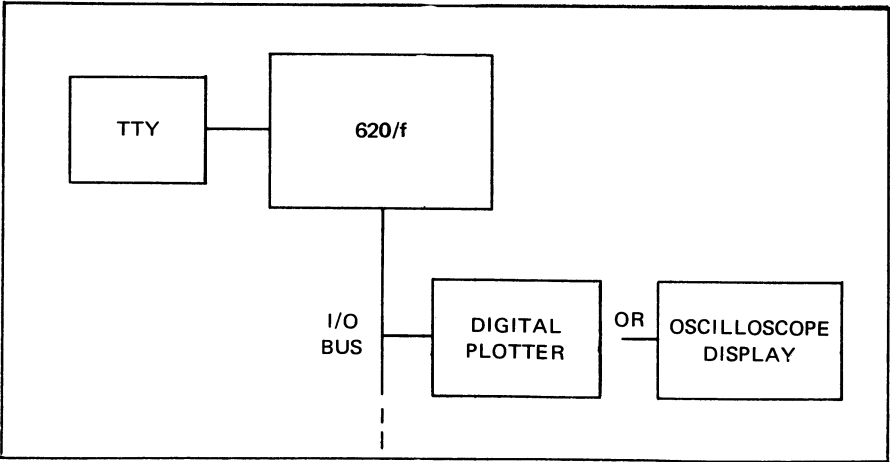
cation lines with differing data rates and formats.

The same equipment can also be used as a preprocessor for a computer center, such as the system illustrated in Figure 5-9. A disc or magnetic tape is used as a data buffer between the communication lines and the major computer. The Varian 620/f relieves the major computer of all control tasks relating to the communication lines and orders the information in a format suitable for direct computer entry.



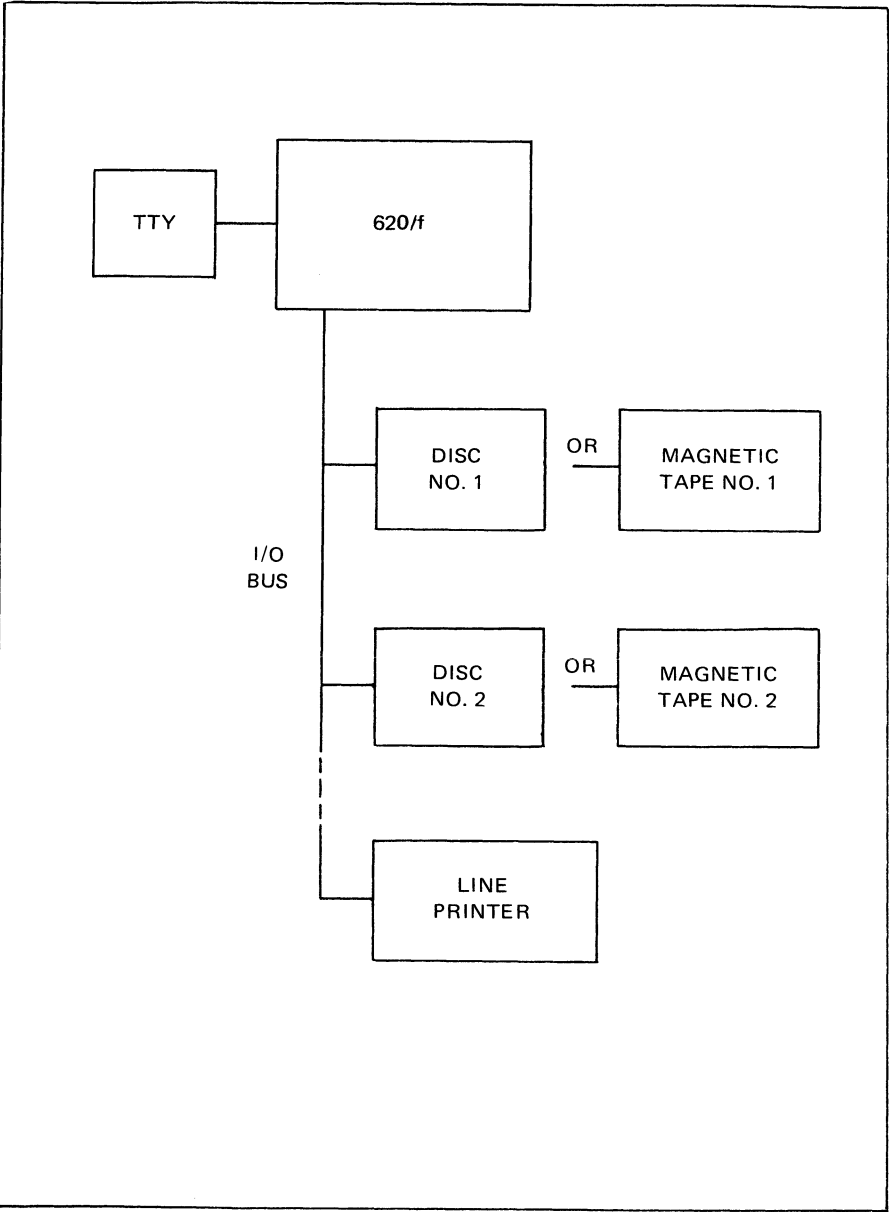
VTII-1230

Figure 5-2. Expanded System for Multiple Users



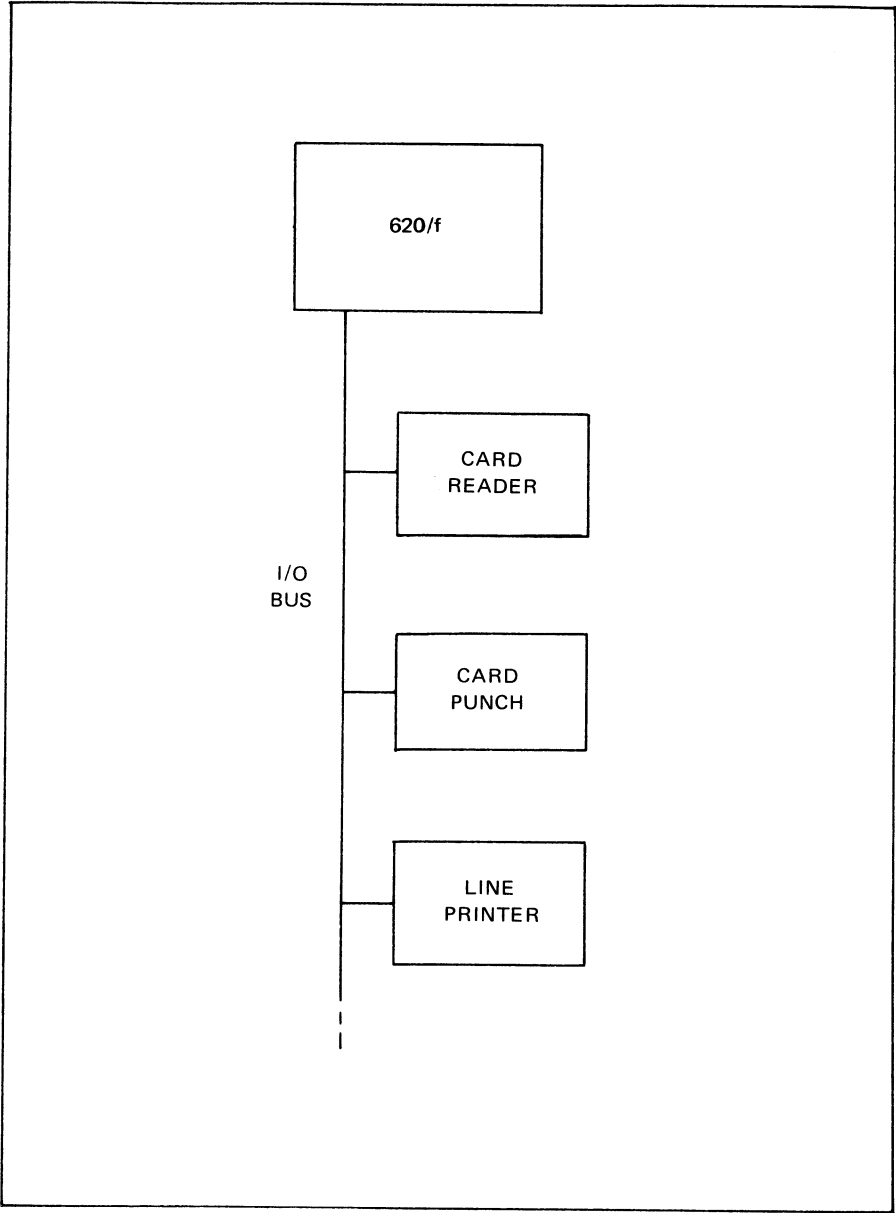
VTII-1231

Figure 5-3. Expanded System with Graphic Outputs



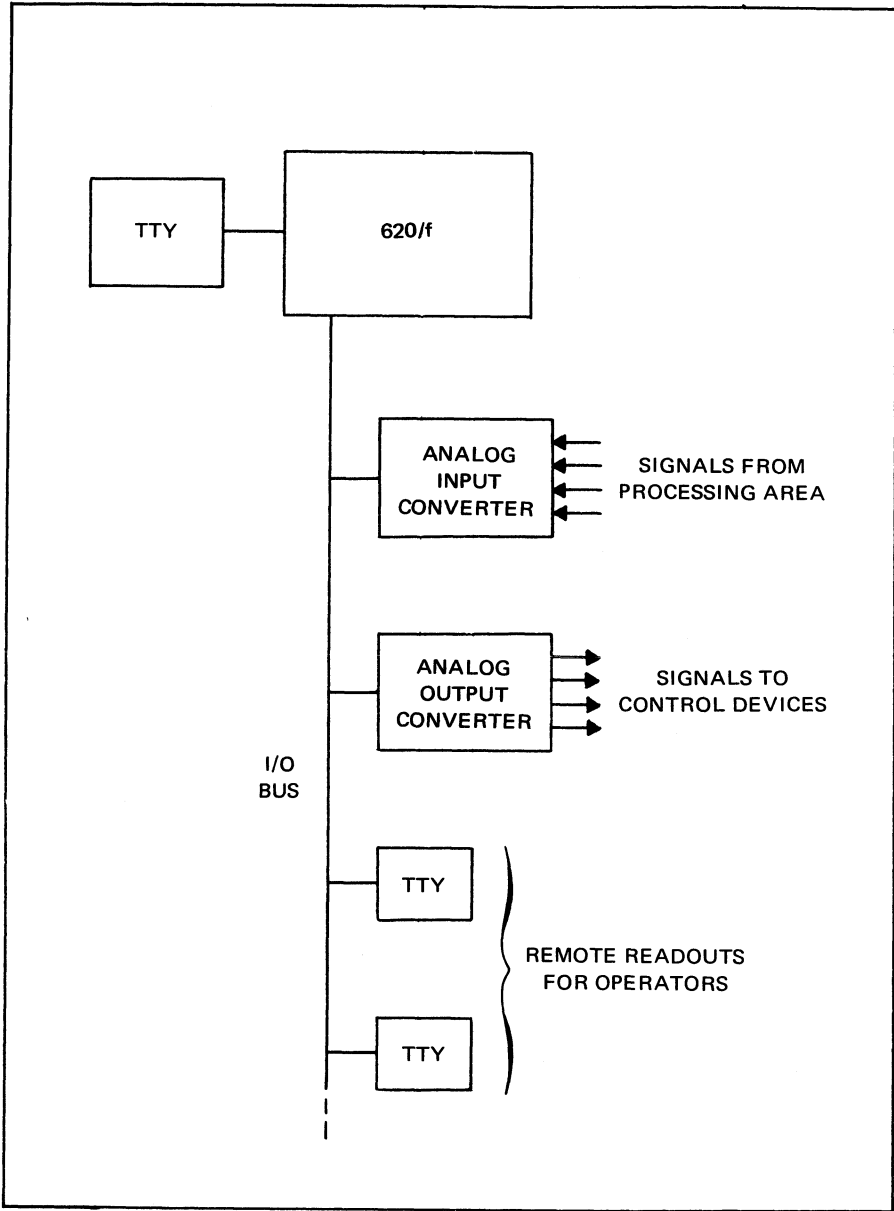
VT11-1232

Figure 5-4. Expanded System for Business Applications



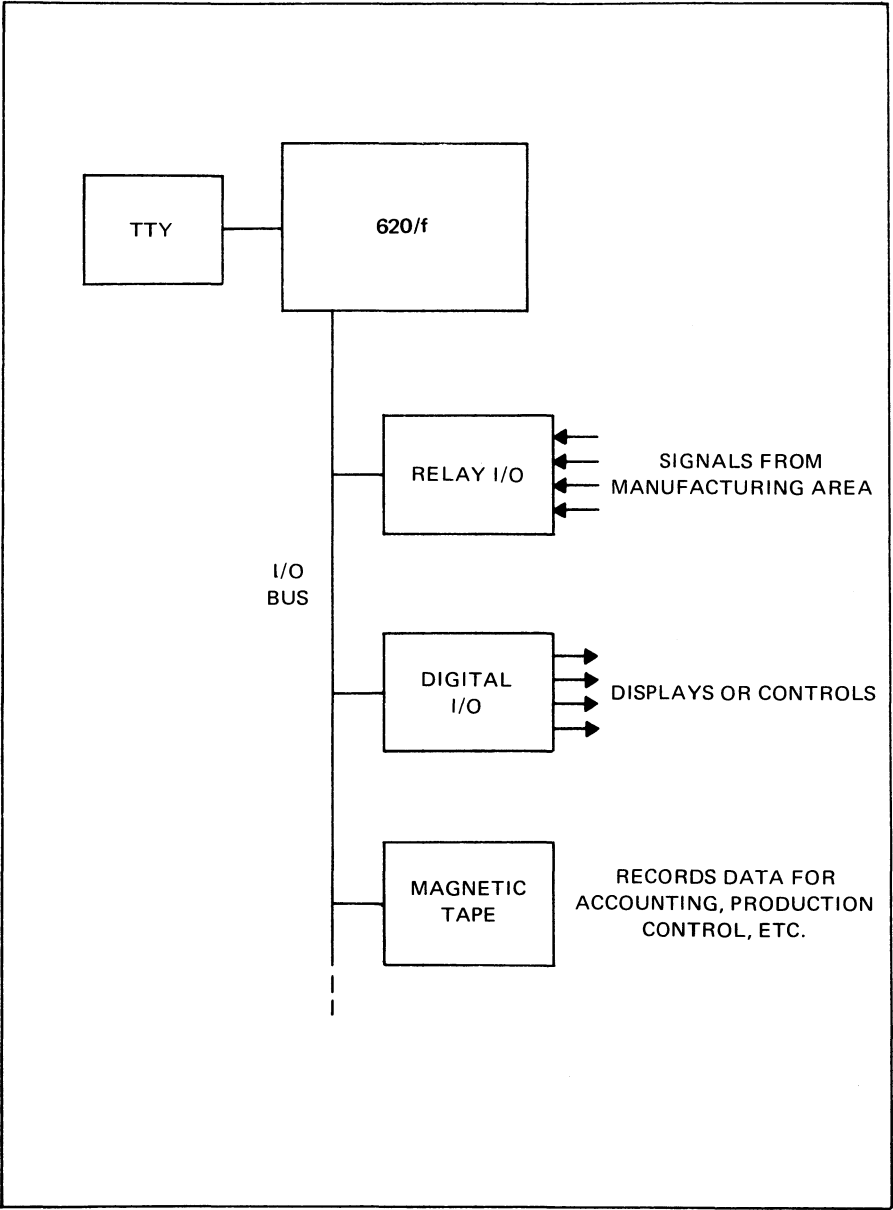
VTII-1233

Figure 5-5. Report Program Generator (RPG IV) System



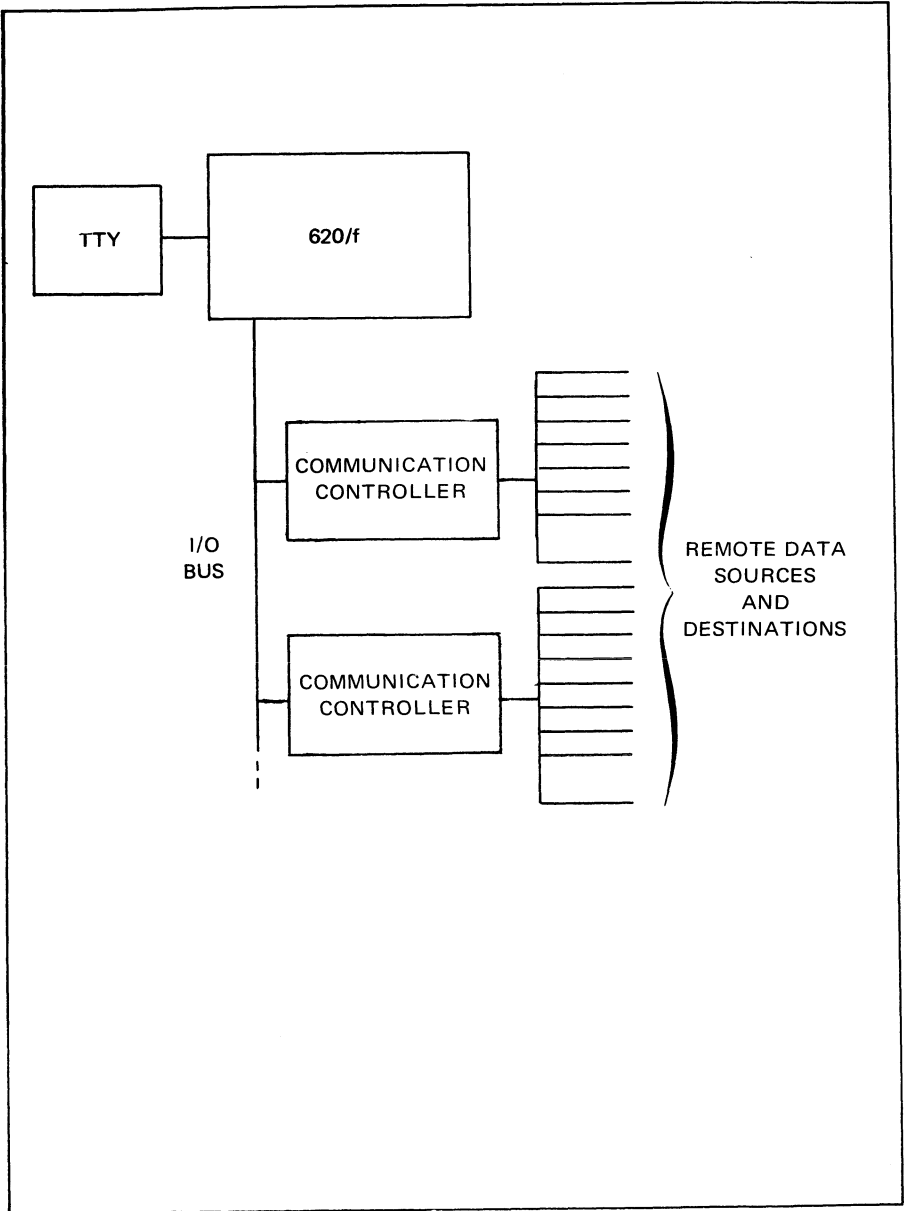
VT11-1234

Figure 5-6. Real-Time Process Control



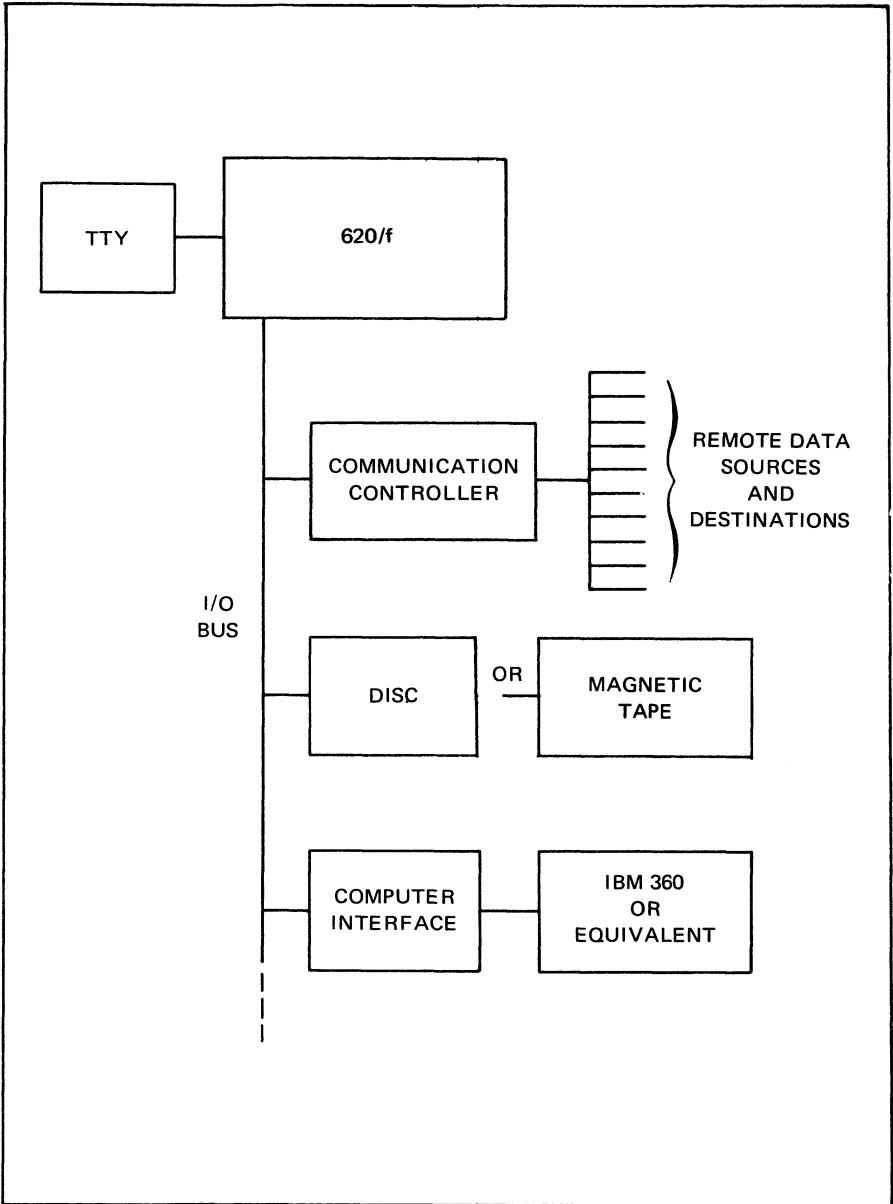
VT11-1235

Figure 5-7. Real-Time Manufacturing Control



VT11-1236

Figure 5-8. Data-Communication Switching Center



VT11-1237

Figure 5-9. Preprocessor for Computer Center

SECTION 6 – VARIAN 620/f SPECIFICATIONS

Type	General-purpose, parallel-operation, digital computer.
Memory	A 3W/3D magnetic core memory with a 16-bit word length, 750 nanoseconds full cycle time, 400 nanoseconds access time, 4,096 words basic and expandable to 32,768 words in 4,096 increments. Control signals between the CPU and memory permit an asynchronous interface.
Word Length	Sixteen bits.
Machine Cycle Speed	750 nanoseconds.
Arithmetic	Binary, two's complement notation.
Data	Fixed point, 16-bit format.
Operational Registers	A Register: 16-bit accumulator. B Register: 16-bit accumulator (least significant half of double-length accumulator) or index register. X Register: 16-bit index register. P Register: 15-bit program counter.
Front Panel Interrupt	This switch can be used to interrupt the computer in the run mode.
Computer Options	Memory protect Buffer interlace controller (BIC) Optional Instruction Set: Hardware multiply/divide (M/D) Bit test (BT) Skip if register equal (SRE) Power failure/restart (PF/R) Real-time clock (RTC) Automatic bootstrap loader (ABL) Priority interrupt module (PIM) (increments of eight) High-speed priority memory access (PMA) Teletype (TTY) Controller

I/O System

Operates asynchronously with the CPU, can accommodate up to ten devices on the bus. It can be expanded to include automatic block transfer (PMA), multilevel priority interrupt (PIM), and cycle-stealing data transfers (BIC).

**I/O Program Control
Instructions**

Data Transfer In:

Single-word, nonaddressing

Double-word, addressing

Data Transfer Out:

Single-word, nonaddressing

Double-word, addressing

External Control:

Single-word, nonaddressing

Program Sense:

Double-word, addressing

SECTION 7 – VARIAN 620/f MODEL NUMBERS

Model No.		Prerequisite
620/f -000	Systems Computer with 4096 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply and Priority Interrupt	None
620/f-001	Systems Computer with 8192 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply and Priority Interrupt	None
620/f-002	Systems Computer with 12,288 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply, Priority Interrupt	None
620/f-003	Systems Computer with 16,384 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply, Priority Interrupt	None
620/f-004	Systems Computer with 20,480 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply, Priority Interrupt	None
620/f-005	Systems Computer with 24,576 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply, Priority Interrupt	None
620/f-006	Systems Computer with 28,672 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply, Priority Interrupt	None

model numbers

620/f-007	Systems Computer with 32,768 words of Read/Write Core Memory, 750 nanoseconds cycle time, Console, Programmed I/O Party Line, Direct Memory Access, Power Supply, Priority Interrupt	None
620/f-010	Expansion Chassis I, for up to 8192 words of additional Memory and 6 controller cards	620/f-00X
620/f-011	Expansion Chassis II, for up to 16,384 words of additional Memory and 2 controller cards	620/f-00X
620/f-012	Expansion Chassis III, 24 controller card positions	620/f-00X
620/f-020	4096 words of Read/Write Core Memory	*See Note
620/f-021	8192 words of Read/Write Core Memory	*See Note
620/f-95-5	Expansion Power Supply	620/f-01X
620/f-05	Memory Protect with Privileged Instructions	620/f-00X
620/f-10	Optional Instruction Set, includes Hardware Multiply/Divide and additional instructions	620/f-00X
620/f-12	Priority Memory Access	620/f-00X
620/f-13	Real Time Clock	620/f-00X
620/f-14	Power Failure/Restart	620/f-00X
620/f-15	Automatic Bootstrap Loader	620/f-00X
620/f-16	Priority Interrupts, 8 multilevel (maximum 64 levels)	620/f-00X or -01X
Teletypes		
620/f-06	ASR-33 (first) and Controller, keyboard/printer with 10 cps paper tape reader and punch, full duplex	620/f-00X
620/f-06-A	Spare Teletype Controller, ASR-33/35 or KSR-35 (620/f-06, -07 or -08 only)	None
620-06-B	Spare ASR-33 (620-06 only), without controller	None
620-06-C	Spare ASR-33 (620-06-E only), 230 Volt, 50 Hz, without controller	None

620-06-D	ASR-33 and Controller, used with Teletype Buffer Board for expansion after the first TTY	620-09, 620/f-00X or 620/f-01X or 620i/L-00 or 620i/L-01
620/f-06-E	ASR-33 (first) and Controller, 230 Volts, 50 Hz	620/f-00X
620-06-F	ASR-33 and Controller, used with Teletype Buffer Board for expansion after the first TTY, 230 Volt, 50 Hz	620-09, 620/f-00X or 620/f-01X or 620/i/L-00 or 620i/L-01
620/f-07	KSR-35 (first) and Controller, heavy duty 10 cps keyboard/printer, full duplex	620/f-00X
620-07-A	Spare KSR-35 (620-07 only), without controller	None
620-07-B	Spare KSR-35 (620-07-D only), 230 Volt, 50 Hz, without controller	None
620-07-C	KSR-35 and Controller, used with Teletype Buffer Board for expansion after the first TTY	620-07, 620/f-01X or 620/f-00X or 620i/L-00 or 620i/L-01
620/f-07-D	KSR-35 (first) and Controller, 230 Volt, 50 Hz	620/f-00X
620-07-E	KSR-35 and Controller, used with Teletype Buffer Board for expansion after the first TTY, 230 Volt, 50 Hz	620-09, 620/f-00X or 620/f-01X or 620i/L-00 or 620i/L-01
620/f-08	ASR-35 (first) and Controller, Heavy Duty keyboard/printer with 10 cps paper tape reader and punch, full duplex	620/f-00X
620-08-A	Spare ASR-35 (620-08 only), without controller	None
620-08-B	Spare ASR-35 (620-08-D only), 230 Volt, 50 Hz, without controller	None
620-08-C	ASR-35 and Controller, used with Teletype Buffer Board for expansion after the first TTY	620/f-09, 620/f-00X or 620/f-01X or 620i/L-00 or 620i/L-01
620/f-08-D	ASR-35 (first) and Controller, 230 Volt, 50 Hz	620/f-00X

model numbers

620-08-E	ASR-35 and Controller, used with Teletype Buffer Board for expansion after the first TTY, 230 Volt, 50 Hz	620-09, 620/f-00X or 620/f-01X or 620i/L-00 or 620i/L-01
620-09	Teletype Buffer Board, drives up to 7 TTY Controllers	620/f-00X or 620/L-00 or 620/i-01

General Purpose Interfaces

620-20	Buffer Interlace Controller, Block transfer Supervisor for automatic data transfers for up to 10 peripheral controllers	620/i-01 or 620/f-00X or
620-80	Buffered I/O Controller, general purpose interface, 8 sense lines, 8 control pulses, 16-bit output register, 16-bit input register, one of 8 pulse widths available: 5-20, 20-73, 73-300 μ sec or .27-1.1, 1-4, 3.5-13, 12-50, 40-90 msec	620/i-01, 620/f-00 or 620/L-00
620-81	Digital I/O Controller, 8 sense lines, 8 control pulses	620/i-01 or 620/f-00 or 620/L-00

Accessories

620/f-92-0	I/O Cable consisting of a cable of optional length (5' increments to 20') with 100 pin male card edge connectors	None
620/f-92-1	External Chassis I/O Cable Adaptor — 620/f I/O connector to 620/i type connector	None
620/f-92-3	100 Pin Chassis Mount Connector Set — female connector set assemblies and mounting hardware	None
620/f-92-4	18 Pin Chassis Mount Connector for PIM IN — male connector with all necessary hardware for mounting, wiring, and keying (for interrupt cable)	None
620/f-92-5	18 Pin Chassis Mount Connector for PIM OUT — female connector with all necessary hardware for mounting, wiring, and keying (for interrupt cable)	None
620/f-92-6	Interrupt Cable consisting of a cable of optional length (5' increments to 20') with a 44 pin edge board connector and 18 pin female connector	None

620-92-8	44 Pin Edge Board Connector and Hood Assembly	None
620/f-92-9	Interrupt Cable consisting of cable of optional length (5' increments to 20') with 18 pin female connector and 18 pin male connector	None
620/f-92-14	Power Supply Sequencing Cable, 620/f to 620/i, 5 feet	None
620/f-92-15	PMA Cable and Connectors (2) and chassis mount connector with mounting hardware, 5 feet	None
620/f-92-16	Male Connector for I/O Bus, mate to 620/f-92-3	None
620/f-92-17	Male and Female Connectors for Interrupt Cable, mates to 620/f-92-4 and 620/f-92-5	None
620/f-92-18	BIC Cable and Connectors including cable of optional length (5' increments to 20'), 44 pin edge board connector, and 18 pin male connector	None
620/f-92-19	BIC Cable and Connectors including cable of optional length (5' increments to 20'), 16 pin female connector, and 18 pin male connector	None
620/f-92-20	BIC Connector Set including one male and one female 18 pin chassis mount connectors	None
620/f-92-21	BIC Connector Set including one male and one female 18 pin cable connectors for mates to 620/f-92-20	None
620/f-92-7	Extender Board for 620/f controller cards	None
620/f-92-7A	Extender Board for 620/f CPU module cards	None
620/f-92-7B	Extender Board for CPU and Memory Trays	None
620-92-10	DM 135-0 Multi-usage socket board	None
620-92-11	DM 135-1 Multi-usage socket board kit consisting of a board, 54 individual 14 pin sockets, 6 individual 16 pin sockets (Sockets not mounted on board)	None
620-92-12	DM 135-2 Multi-usage socket board kit consisting of a board, 110 individual 14 pin sockets (Sockets not mounted on board)	None

model numbers

620-92-13	DM 135-3 Multi-usage socket board kit consisting of a board, 156 individual 14 pin sockets, 24 individual 16 pin sockets (Sockets not mounted on board)	None
-----------	---	------

Spares

620/f-95-1	Spares Kit, 620/f Modules	None
620/f-95-2	Spares Kit, 620/f Components	None
620/f-95-3	Memory Spares Kit (4K), including 4K core stack, inhibit-sense module, driver-switch module, and timing and control module	None
620/f-95-4	Memory Spares Kit (8K), including 8K core stack, inhibit-sense module, driver-switch module, and timing and control module	None

Cabinets

620-90	19-inch Cabinet: 30 inches deep, 63 inches high, includes side panels, cooling unit and mounting of standard components	None
620-90-A	19-inch Cabinet: 30 inches deep, 63 inches high, includes side panels, cooling unit, casters and mounting of standard components	None

Card Equipment

620-25	Card Reader & Controller, 300 cards per minute	One U slot
620-27	Card Punch & Controller, 35 cards per minute	One U slot

Magnetic Tape

620-30	Magnetic Tape Unit & Controller, 9-track, 800 bpi, 25 ips, single density, includes control for up to four (4) 9-track magnetic tape units	Two U slots
620-300	Magnetic Tape Unit Slave, 9-track, 800 bpi, 25 ips, single density	620-30
620-31A	Magnetic Tape Unit and Controller, 7-track, Dual Density (200/556 bpi), 25 ips, includes control for up to four (4) 7-track magnetic tape units	Two U slots

620-310A	Magnetic Tape Unit Slave, 7-track, Dual Density (200/556 bpi), 25 ips	620-31A
620-31B	Magnetic Tape Unit and Controller, 7-track, Dual Density (200/800 bpi), 25 ips, includes control for up to four (4) 7-track magnetic tape units	Two U slots
620-310B	Magnetic Tape Unit Slave, 7-track, Dual Density (200/800 bpi), 25 ips	620-31B
620-31C	Magnetic Tape Unit and Controller, 7-track, Dual Density (556/800 bpi), 25 ips, includes control for up to four (4) 7-track magnetic tape units	Two U slots
620-310C	Magnetic Tape Unit Slave, 7-track, Dual Density (556/800 bpi), 25 ips	620-31C

Rotating Memory

620-37	Disc Memory and Controller, moving head, capacity 1170K 16 bit words, transfer rate 92K words per second, average access time on track 20 ms. Controller controls master and up to 3 slave drives	One U slot 620-20
620-37A	Disc slave transport, moving head, capacity 1170K 16 bit words, transfer rate 92K words per second, average access time on track 20 ms.	620-37
620-38A	Disc Memory & Controller, fixed head, average access time 17 ms, transfer rate 73.3 KHz words, capacity 30K words, 16 tracks	One U slot 620-20
620-38B	Disc Memory & Controller, fixed head, average access time 17 ms, transfer rate 73.3 KHz words, capacity 61K words, 32 tracks	One U slot 620-20
620-38C	Disc Memory & Controller, fixed head, average access time 17 ms, transfer rate 73.3 KHz words, capacity 123K words, 64 tracks	One U slot 620-20
620-39	Disc Memory & Controller, moving head, control for up to two drives, average access on track 20 ms, transfer rate 42 KHz words, capacity 585K words, IBM 2315 disc pack	One U slot 620-20

model numbers

620-39A	Disc slave, moving head, average access on track 20 ms, transfer rate 42 KHz words, capacity 585K words, IBM 2315 disc pack	620-39
620-40	Disc Memory and Controller, moving head, control for up to four drives, average access on track 12.5 ms, transfer rate 80K words per second, capacity 3.625 million 8-bit bytes, IBM 1316 disc pack	Two U slots 620-20
620-41	Disc Memory and Controller, moving head, control for up to four drives, average access on track 12.5 ms, transfer rate 80K words per second, capacity 7.25 million 8-bit bytes, IBM 1316 disc pack	Two U slots 620-20
620-41A	Disc slave, moving head, average access on track 12.5 ms, transfer rate 80K words per second, capacity 7.25 million 8-bit bytes, IBM 1316 disc pack	620-41
620-44	Drum Memory and Controller, average access time 8.7 ms, approx. transfer rate 106K words per second, 16 tracks, capacity 30K words	One U slot 620-20
620-45	Drum Memory and Controller, average access time 8.7 ms, approx. transfer rate 106K words per second, 32 tracks, capacity 61K words	One U slot 620-20
620-46	Drum Memory and Controller, average access time 8.7 ms, approx. transfer rate 106K words per second, 64 tracks, capacity 123K words	One U slot 620-20
620-47	Drum Memory and Controller, average access time 8.7 ms, approx. transfer rate 106K words per second, 128 tracks, capacity 246K words	One U slot 620-20
620-48	Drum Memory and Controller, average access time 8.7 ms, approx. transfer rate 106K words per second, 256 tracks, capacity 491K words	One U slot 620-20
620-49	Drum Memory and Controller, average access time 8.7 ms, approx. transfer rate 106K words per second, 512 tracks, capacity 983K words	One U slot 620-20

Paper Tape

620-51	Paper Tape Reader & Controller, 300 cps	One PC slot
620-51		
620-51A	Paper Tape Reader & Controller, 150 cps	One PC slot
620-54	Paper Tape Punch & Controller, 75 cps. Table Top or chassis slide (-RM) for 19" rack mounting. For operation at 50 to 100 Hz @ 110/127/220/240 VAC	One PC slot
620-55	Paper Tape System, includes time-share controller, 150 cps reader, 75 cps punch	One PC slot
620-55A	Paper Tape System, includes time-share controller, 300 cps reader, 75 cps punch	One PC slot

Displays and Printers

620-72	Digital Plotter, 300 steps per second, 0.01" step size, other sizes are available	One U slot
620-73B	Oscilloscope Display (Tektronix Model 611), 11 inch storage scope	One PC slot 620-88
620-74	Printer/Plotter and Controller, Statos® 21, Model 2110, 167 steps per second (equivalent to 1000 alphanumeric lines per minute) in asynchronous mode, 0.0125" step size, 80 styli per inch	
620-74A	Printer/Plotter with pedestal and Controller, Statos® 21, Model 2111, same characteristics as 620-74 plus a character generator for 835 steps per second (equivalent to 5000 alphanumeric lines per minute) in synchronous mode	
620-77	Line Printer, 245 to 1100 lpm, 132 columns, segmented buffered	One U slot

Digital Controllers

620-80	Buffered I/O Controller, general purpose interface, 8 sense lines, 8 control pulses, 16-bit output register, 16-bit input register. One of 8 different pulse widths available (5-20, 20-73, 73-300 usec; .27-1.1, 1-4, 3.5-13, 12-50, 40-90 msec)	One U slot
--------	--	------------

model numbers

620-81	Digital I/O Controller, 8 sense lines, 8 control pulses	One PC slot
620-83-1	Relay Contact I/O Module, 16 mercury wetted contact outputs, 50 B A resistive, 3 A or 400 V maximum	One U slot
620-83-2	Relay Contact I/O Module 16 contact points, voltage levels, 12 B A resistive, 1/2 A or 200 V maximum	One U slot
620-83-3	Relay Contact I/O Module 16 mercury wetted contact outputs and 16 contact inputs, 50 V A resistive, 3 A or 400 V maximum	One U slot

Analog-to-Digital Converters

620-85A	Analog-to-Digital Converter: 13-bit binary, ± 10 volts F.S. input, 55 KHz throughput (max), Sample and Hold input and programmable timer	One PC slot 620-88
620-850	Analog Input System: ADC with 13-bit binary, 55 KHz throughput, Sample and Hold, programmable timer. Multiplexer with 16 single-ended channels expandable to 256 channels.	Two PC slots 620-88
620-851	Analog Input System: ADC with 13-bit binary, 55 KHz throughput, Sample and Hold, programmable timer. Multiplexer with 16 differential channels expandable to 256 channels	Two PC slots, 620-88

Multiplexers

620-860	Multiplexer Module, includes multiplexer control and 16 single-ended input channels. Input is ± 10 volts for full scale and 100 meg. ohms impedance	One PC slot, 620-88
620-860,^	Multiplexer Expansion Module. 16 single-ended input channels	One PC slot, 620-860, 620-861 or 620-850, 620-851
620-861	Multiplexer Module, includes multiplexer control and 16 differential input channels. Input is ± 10 volts for full scale and 100 meg. ohms impedance	One PC slot, 620-88

620-861A	Multiplexer Expansion Module. 16 differential input channels.	One PC slot, 620-860, 620-861 or 620-850, 620-851
----------	---	---

Digital-to-Analog Converters

620-870	DAC Module, one 10-bit binary DAC, ± 10 volts, full scale output at ± 5 ma (max), includes DAC control	One PC slot, 620-88
620-870A	DAC Module, two 10-bit binary DAC, includes DAC control	One PC slot, 620-88
620-870B	DAC Expansion Module, two 10-bit binary DAC	One PC slot, 620-870 or 620-870A
620-871	DAC Module, one 12-bit binary DAC, ± 10 volts full scale output at ± 10 ma (max), includes DAC control	One PC slot, 620-88
620-871A	DAC Module, two 12-bit binary DAC includes DAC control	One PC slot, 620-88
620-871B	DAC Expansion Module, two 12-bit binary DAC	One PC slot, 620-871 or 620-871A
620-872	DAC Module, one 14-bit binary DAC, ± 10 volts full scale output at ± 10 ma (max), includes DAC control	One PC slot, 620-88
620-872A	DAC Module, two 14-bit binary DAC, includes DAC control	One PC slot, 620-88
620-872B	DAC Expansion Model, two 14-bit binary DAC	One PC slot, 620-872 or 620-872A
620-873	DAC Model, one 10-bit binary DAC, ± 10 volts full scale output at ± 5 ma (max), and one 12-bit binary DAC ± 10 volts full scale output at ± 10 ma (max). Includes DAC control	One PC slot, 620-88
620-874	DAC Module, one 10-bit binary DAC ± 10 volts full scale output at ± 5 ma (max), and one 14-bit binary DAC ± 10 volts full scale output at ± 10 ma (max). Includes DAC control	One PC slot, 620-88
620-875	DAC Module, one 12-bit binary DAC and one 14-bit binary DAC. Both have ± 10 volts full scale output at ± 10 ma (max). Includes DAC control	One PC slot, 620-88

model numbers

Power Supply

620-88	Analog Power Supply: Input voltage 115/230 VAC $\pm 10\%$ 47 Hz to 63 Hz. Input current 1.6 RMS F.L. Outputs: +5 VDC at 5 A, ± 15 VDC at 1 A, ± 20 VDC at 250 ma, +24 VDC at 500 ma.	One PC slot (power distribution) Supply mounts on front or near cabinet rails
--------	--	---

Data Communication

620-60	Communication Controller. Sixteen line multiplexer which controls up to four 620-61's (sixteen 103 type data sets), allows for programmed bit assembly/disassembly of characters	620/i-01
620-61	Communication Line Control Module. Provides interface between Communication Controller (620-60) and four Bell 103 type or equivalent data sets	620-60, 620-61A (2)
620-61A	Data Set Cable, 25 feet, interconnects two data sets and a Line-Control Module (620-61)	
620-62	Automatic Call Unit Controller. The ACU controller provides program control of the 801 (A/C) Data Auxiliary Set and permits dialing any telephone number in the switched telephone network. Includes a 10-foot data set cable	1 U I/O slot
620-65A	Data Set Controller — Bell 201 type or equivalent full/half duplex synchronous operation. Speed up to 4800 baud, single character buffered, software sync recognition, code transparency, automatic answer. Includes a 10-foot data set cable	1 U I/O slot
620-65B	Dual 620-65A's Data Set Controllers. Includes two 10-foot data set cables	1 U I/O slot
620-65C	Data Set Controller — Bell 201 type or equivalent, full/half duplex synchronous operation. Speed up to 50,000 baud, double character buffering, hardware line synchronization (sync characters may be changed under software control), code transparency, automatic answer. The 620-65C may be operated in conjunction with the DMA channel and Buffer Interface Controller (620-20). Includes a 10-foot data set cable	1 U I/O slot

620-66A	Data Set Controller — Bell 103 type or equivalent, full/half duplex asynchronous operation, speed up to 300 Baud, 9, 10, or 11 bit character format, with automatic answering capability. Includes a 10-foot data set cable	1 U I/O slot
620-66B	Dual 620-66A's Data Set Controllers. Includes two 10-foot data set cables.	1 U I/O slot
620-67	Data Set Controller — Bell 202 type or equivalent, full/half duplex asynchronous operation, speed up to 2000 Baud, 9, 10, or 11 bit character format, Reverse Channel, automatic answer, hardware parity checking, data transfers are under program control or the Buffer Interface Controller option may be utilized. Includes a 25-foot data set cable	1 U I/O slot
620-68	Communication Controller. The 620-68 includes a 64 channel Multiplexer, line control buffer storage for 16 channels, and chassis space for: an additional line control buffer for 16 channels (1-Line Storage Module 620-68A) and line control modules for 32 channels. When expanding to more than 32 channels, a 620-68B expansion chassis is required in addition to Line Storage modules and Line Control Modules	620f
620-68A	Line Storage Module. 16 channel line buffer, line control modules for 16 channels may be connected to a Line Storage Module. First 620-68A will fit in the 620-68.	620-68 or 620-68B
620-68B	Line Expansion Chassis and Power Supply Expansion for channel 33 to 64	620-68
620-68-1	Line Control Module. With a RS232B interface, 4 channels of full/half duplex asynchronous/synchronous operation	620-68 or 620-68B
620-68-2	Line Control Module. With a CCITT interface, 4 channels of full/half duplex asynchronous/synchronous operation	620-68 or 620-68B
620-68-3A	Line Control Module. For interfacing to a two-wire discrete communication system, 8 channels of half-duplex asynchronous operation	620-68 or 620-68B

model numbers

620-68-3B	Line Control Module. For interfacing to a two-wire discrete communication system, 4 channels of half-duplex asynchronous operation	620-68 or 620-68B
620-68-4A	Line Control Module. For interfacing to a relay communication system, 8 channels of full/half duplex synchronous operation	620-68 or 620-68B
620-68-5	Line Control Module. With a MIL 188B interface, 2 channels of full/half duplex synchronous operation	620-68 or 620-68B
620-92-2A	Cable from a Line Control Module to two Modems 10 feet each with mating Modem connectors	
620-92-2B	Cable from a Line Control Module to two Modems 25 feet each with mating Modem connectors	
620-92-2C	Cable from a Line Control Module to two Modems 50 feet each with mating Modem connectors	

SECTION 8 – CENTRAL PROCESSOR UNIT

The Varian 620/f computer has three major functional sections, as shown in Figure 8-1:

1. The Central Processor Unit (CPU) section.
2. The Memory section.
3. The Input/Output section.

The memory is described in detail in Section 9; the external I/O structure, in Sections 11 and 12. The CPU and its interface with the memory and I/O structure are detailed in the following paragraphs.

CPU Section

The CPU section consists of clock circuits, data buses, operation and auxiliary registers, arithmetic unit, control and decoding circuits, and register display and switch circuits.

The bit slice technique is used in structuring the arithmetic unit and CPU registers. Each of the four data loop cards contains four bits of all registers (and arithmetic unit) with their respective input and output gates.

Clock Circuits

Two crystal-controlled clock circuits on the clock card provide basic timing signals. The main clock circuit generates an 18-MHz master clock to the CPU control circuits for memory interface. The main clock circuit also divides the 18-MHz clock by two and applies the resulting 9-MHz clock to the control circuits for state control.

The I/O clock circuit generates an 8.8-MHz master clock for the I/O section.

Buses

The basic computer contains eight buses designated as C, AY, AZ, MB, MD, ABS, AB, and I/O.

The C bus provides the data path and selection logic for routing data from the arithmetic unit to the operation registers (A, B, X, and P), the auxiliary registers (D, I, and L), and register display.

The AY bus routes selected data from the I and R registers and register entry switches.

The MB and MD buses provide data paths to and from memory, respectively.

The ABS bus routes address information into memory from the address (L) register and the D register.

The AB bus provides data paths between the computer and the internal options (RTC, PF/R, MP, and TTY controller).

Operation Registers

The A, B, X, and P registers comprise the Varian 620/f operation registers. The A, B, and X registers can be directly accessed. The P register is indirectly accessible to the programmer through use of the following instruction types.

- a. Jump instructions that modify the program sequence.
- b. Arithmetic/logic instructions in the relative-addressing mode which uses the contents of the P register to modify the operand address.

central processor unit

The outputs of the operation registers are connected to the Y bus and routed to the Y inputs of the arithmetic unit. The inputs to the operation registers come from the C bus, which contains output data from the arithmetic unit.

The A Register is a 16-bit register that serves as the upper half of the accumulator. It is used for:

- a. Accumulating the result of logical operations
- b. Accumulating the result of addition or subtraction
- c. Accumulating the more significant half of the product in multiplication (optional)
- d. Accumulating the remainder in division (optional)
- e. Program-controlled I/O transfers

A shift/rotate (S/R) circuit provides a special data path to implement A and B register shift and rotate operations.

The B Register is a 16-bit register that serves as the lower half of the accumulator. It is used for:

- a. Accumulating the less significant half of the product in multiplication (optional)
- b. Accumulating the quotient in division (optional)
- c. Program-controlled I/O transfers
- d. Indexing (as a second index register)

The X Register is a 16-bit register that permits the indexing of operand addresses without adding time to the execution of

indexed instructions.

The P Register is a 15-bit register that contains the address of the current instruction. It is incremented before each new instruction is fetched.

Auxiliary Registers

The I, R, L, and D registers comprise the auxiliary registers. The D register is in the I/O section (see below). The outputs of the I and R registers are connected to the AZ bus and routed to the AZ input of the arithmetic unit. The inputs to the I and R registers come from the MD bus, which contains memory output data.

The I Register is a 16-bit register that receives each instruction from memory through the MD bus and holds the instruction during its execution. Instructions can be loaded in the I register from the C bus via the control panel register entry switches. The control fields of the instruction word are routed to the decoding section to determine the required timing and control signals. The five least significant bits of the I register are transferred into a shift counter to shift-count-control the shift instructions.

The R Register is a 16-bit buffer register that buffers the arithmetic unit from memory to permit interlaced I/O operations to steal memory cycles. The R register holds the second word of a two-word instruction, the multiplicand in multiplication (optional), and the divisor in division (optional).

The L Register is a 15-bit address register that contains the address of the memory location currently being accessed during either a clear/write or read/restore cycle.

Arithmetic Unit

The arithmetic unit contains the adder and

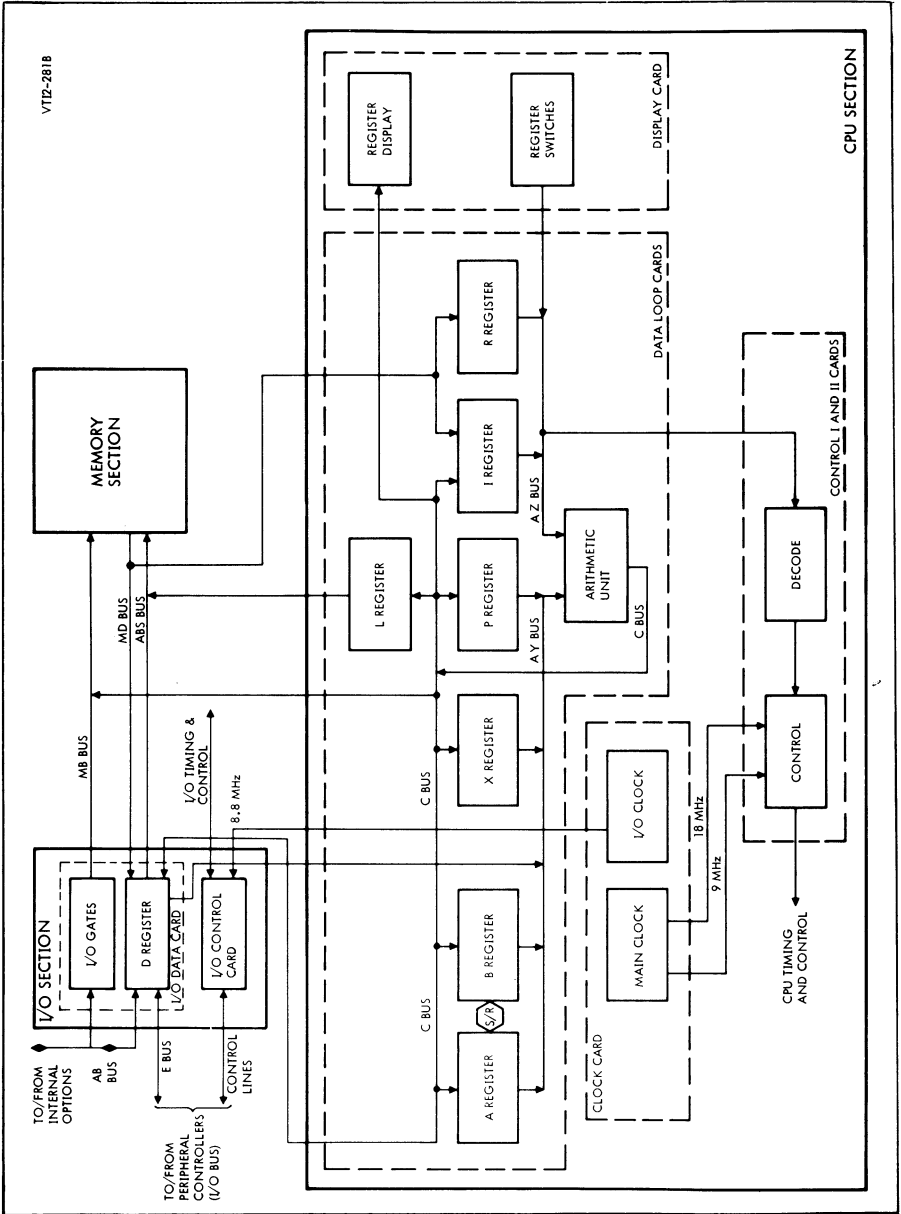


Figure 8-1. Varian 620/f Organization

central processor unit

gating for all operations except shifting. Under the control of the CPU control circuits, the arithmetic unit processes data from the AY and AZ buses and sets the result on the C bus.

Control Circuits

The control circuitry generates timing and control signals using the information received from the I register. The control circuits are contained on the control I and II cards.

Decoding Circuits

The decoding circuitry decodes the bits of the word in the I register used to determine control signal levels generated by the control circuits. The decoding circuits are contained on the control I and II cards.

Register Display and Switch Circuits

This section displays the contents of and loads registers or memory on the Varian 620/f control panel (Section 14). The register display and switch circuits are contained on the display card.

Memory Interface

The expandable, random-access, three-wire/three-dimensional magnetic core memory is used for program and data storage. It operates asynchronously with the CPU and I/O sections.

Instructions from memory are transferred to the control section for execution. Under program control, words can be transferred

from memory to the arithmetic unit, operations register, or I/O bus and, conversely, to memory from the operations registers or the I/O bus.

Words written into memory are transferred on the MB bus; words read out, on the MD bus. Memory addresses are transferred to memory on the unidirectional ABS bus.

A modular master/slave configuration permits economical memory expansion. The slave module (or modules) uses the timing and control circuits of the master memory module.

I/O Interface

Peripheral devices communicate with the computer on the I/O bus. The 16-bit, party-line, bidirectional I/O bus transfers programmed data between peripheral devices and the computer. The I/O bus also permits plug-in expansion of Varian 620/f peripherals. Part of the I/O bus is an E bus for bidirectional data transfer.

Information exchanges with peripheral devices are synchronized by peripheral controllers, which can control one or more similar devices. Each controller and the device (or devices) it controls comprise a peripheral device option.

The I/O section is on two circuit cards. The I/O data card contains the 16-bit D register, which stores E bus I/O information. The I/O control card contains timing and control circuits for program-controlled I/O, cycle-stealing, I/O, and interrupt operations.

SECTION 9 – MEMORY

The Varian 620/f memory system is an expandable, random-access, 3W/3D, magnetic-core memory. The basic package accommodates 4,096 or 8,192 sixteen-bit words. The system is expandable to 32,768 words in 4,096- or 8,192-word increments.

The performance specifications on the Varian 620/f memory system are listed in Figure 9-1. A timing diagram for the system is given in Figure 9-2.

Master Memory System

The master memory system is contained on four printed circuit modules. The modules are stacked (see Figure 15-5, Section 15), with module 1 on the bottom. The modules are designated as follows:

- Module 1: Timing Control and Address and Data Register (TAD)
- Module 2: Drivers and Switches(DS)
- Module 3: Inhibit Driver/Sense Amplifier (IDSA)
- Module 4: Memory Stack Module

Module 1 is attached to a rigid frame that rides on the computer chassis slides, providing for insertion and removal of the memory system within the computer chassis. Three card-edge connectors in module 1 plug into the computer backplane. This is the only electrical connection between the memory system and the computer.

In addition to the card edge connectors, module 1 contains two right-angle connectors which are receptacles for a short PC card. The PC card terminates wires that connect to card-edge connectors on modules 2, 3, and 4. This interconnect wiring arrangement allows each module to be disconnected from the memory system assembly without unsoldering wires.

The four modules are tied together during operation by five through bolts on the forward end (opposite the hinged end). In addition, standoff spacers are located between modules for vibration suppression.

The stack module is a printed-circuit construction, with a core mat on the upper and lower surfaces of the module. The core mats are protected by removable perforated covers. Two thermistors located on the core stack module monitor temperature and regulate core drive current.

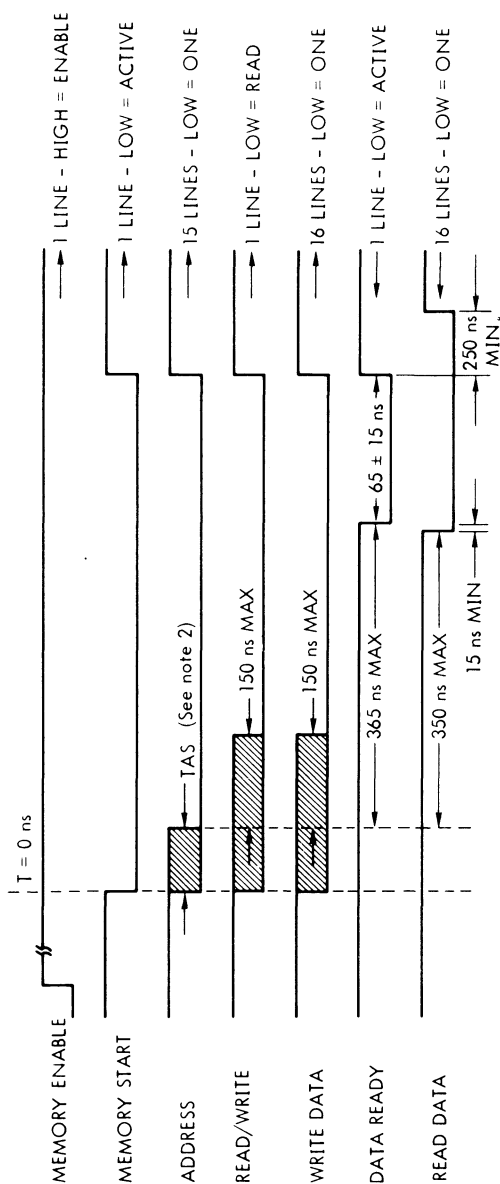
The other three modules (TAD, DS, and IDSA) consist of standard printed circuits. Three of the six fans on the mainframe or expansion chassis provide cooling for the memory system. The cooling system is designed for a maximum 3 degrees C rise above ambient at the exhaust side of board except for an approximate 10 degrees C rise across the resistors mounted on the IDSA module. In a memory expansion configuration, all stacks are within 5 degrees C of each other.

Slave Memory System

The slave memory system is the same as the master memory system except that module

Specification	Description
Timing	Time measured from 50 per cent point of select leading edge*
	Cycle time: 700 nanoseconds**
	Access time: 350 nanoseconds**
	Repetition ratio: one cycle every 700 nanoseconds**
	Write data available from 150 (maximum) to 350 nanoseconds (minimum)**
Typical Tests	Address available at 0 nanosecond, settled out (option for a fixed delay in address of 50 nanoseconds, maximum, available)
	Memory start pulse width: 40 nanoseconds (minimum)
	Patterns: All ones All zeros Worst case Address Random
	All patterns may be complemented by word or bit
	Read/restore mode: All cycles error-checked
Operating Limits	Memory is capable of:
	a. Operating indefinitely at any address
	b. Operating in the burst mode, i.e., any series of memory cycles interspersed by nonoperating time
	X/Y current: ± 5 percent minimum around optimum point
	Inhibit current: ± 5 percent minimum around optimum point
	Logic power supply voltages can be biased ± 5 percent from nominal
*Signals measured at the interface connector of the memory system.	
**These times are valid only when the address is available at time 0 (see figure 9-2).	

Figure 9-1. Varian 620/f Memory System Performance Specifications



NOTES:

1. Cycle time is 700 nanoseconds plus the fixed address settling time (TAS) of 50 nanoseconds maximum.

VTII-1147

Figure 9-2. Memory Timing

memory

1 contains no timing control circuits and registers. The slave-memory module 1 consists of a short PC card containing three card-edge connectors for interconnection with the master memory. In addition, wires from modules 2, 3, and 4 are soldered to the short PC card.

Timing Control and Address and Data Registers (TAD)

The TAD module contains circuitry that includes timing control, address register, address decoder, data register, line drivers, and current sources.

The timing control is formed by two delay lines and TTL gates to furnish all the timing and control signals for the memory system. The internal system timing and control signals supplied to the other circuits consist of address register load, data register load and reset, and timing for the switch/drivers and inhibit drivers. The interface control signals include busy and data ready.

The 15-bit jam-transfer address register accepts the address from the CPU at the beginning of every cycle. Four address decoders decode 12 bits of the address register. The decoded outputs are fed to the switch/driver circuits. The remaining three address bits are used for phase control and stack selection.

A 16-bit data register accepts data from the CPU in the write cycle and from the sense amplifier (sense/inhibit module) in the read/restore cycle. The outputs of the data register are supplied to the line drivers to deliver the output data and to the inhibit drivers to form the data recirculation loop.

The TAD module also contains two current sources to drive X and Y lines. The current source is a series-type current regulator with the level adjustable by the potentiometer on the module. The thermistor controls the

current level with respect to the stack temperature to provide the optimum core drive.

Drivers and Switches (DS)

The DS module contains 32 pairs of switches and drivers. The switch/drivers provide current switches to drive the core matrix (64-by-64 configuration). Each of the 64 x (or Y) drive lines is decoded by an 8-by-8 X (or Y) switch matrix. For every combination of address inputs, one decoded output is enabled; thus, only one switch/driver pair is enabled during a cycle. A read or write command from the timing and control circuit determines the time at which the selected pair is to provide source or sink current.

Inhibit Driver/Sense Amplifier (IDSA)

The IDSA module contains 32 inhibit drivers and 32 sense amplifiers. Each half of the inhibit drivers and sense amplifiers is associated with a 4,096 sixteen-bit word core mat.

The inhibit drivers, controlled by the inhibit timing, receive memory word data from the data register to provide inhibit current for a data bit of zero and no inhibit current for a one during the write half-cycle.

The sense amplifier compares the core output from the sense line to the predetermined threshold voltage derived from the threshold reference. The output of the sense amplifier is strobed by the sense strobe signal and buffered by an open collector gate to provide a wired-OR capability for the memory expansion.

One voltage regulator on this module provides a threshold reference for the 32 sense amplifiers.

Memory Stack Module

The memory stack module consists of a diode decoding matrix and a magnetic core

array. The stack is organized in 3W/3D, 8,192 sixteen-bit words. An optional configuration of 4,096 sixteen-bit words is available. The anticoincident core-driving technique is used to reduce the number of switch/drivers required to operate a typical 8,192-word stack. The 64 X and 64 Y drive lines address 4,096 words for coincident and another 4,096 for anti-coincident.

In the 8,192-word stack, there are thirty-two 4,096-word core mats. Each mat is equipped with an individual sense/inhibit line.

CPU/Memory Interface

The memory interface is TTL-compatible. The address and data lines are unidirectional buses. Since the buses can exceed 10 inches long, a driver/receiver scheme is required for their implementation.

The memory is a single-port and operates in a request (memory start pulse) response (data ready pulse) mode with the CPU, permitting asynchronous operation between it and the CPU.

The CPU requests memory by raising memory start. After a decode delay, the memory cycle begins. When memory samples the data or has data on the bus, it transmits a data ready pulse. The trailing edge of this pulse clears the memory start, memory address, and write data signals at the CPU. The leading edge of the data ready signal strobes data from memory into the CPU. The memory start (MS) signal is logically ANDed with memory busy to initiate a new memory cycle.

Figure 9-2 shows the interface waveforms for the Varian 620/f memory system. Critical timing relationships involve memory bus transmission times, which must take into account skew between data and control

paths. The time from the read memory data (memory access) until the leading edge of data ready should be sufficient to ensure that data are stable at the CPU.

Memory Protect

The memory system uses the write state of the read/write line to strobe write data into the register. This occurs prior to the leading edge of the data ready pulse. If Memory Protect is violated, the CPU changes this line to the read state, and the contents of memory are restored. The read/write control can be changed up to 150 nanoseconds after the address lines are settled (TAS).

Wrap-Around Addressing

Wrap-around addressing is available (upon request) as a standard feature in the Varian 620/f memory system. This feature automatically prevents the CPU from trying to address data at nonexistent memory locations. For example, if the P register in a computer with 8K of memory indicates that the address of the next instruction to be executed is 8,192, wrap-around addressing causes this instruction to be executed from address 0 instead of the nonexistent 8,192 address.

Likewise, the instruction in address 8,193 would be executed from address 1. In this same manner, any nonexistent memory address is automatically redirected to the corresponding existing memory location. Without the wrap-around feature, data read from a nonexistent memory location are zero and data written in are lost.

Wrap-around addressing is implemented with three jumper connections on module 1 of a master memory system. The jumpers connect to address lines of the three most significant address bits that select a particular 4K memory increment. If these jumpers are connected in various combinations, wrap-around addressing of 4K, 8K,

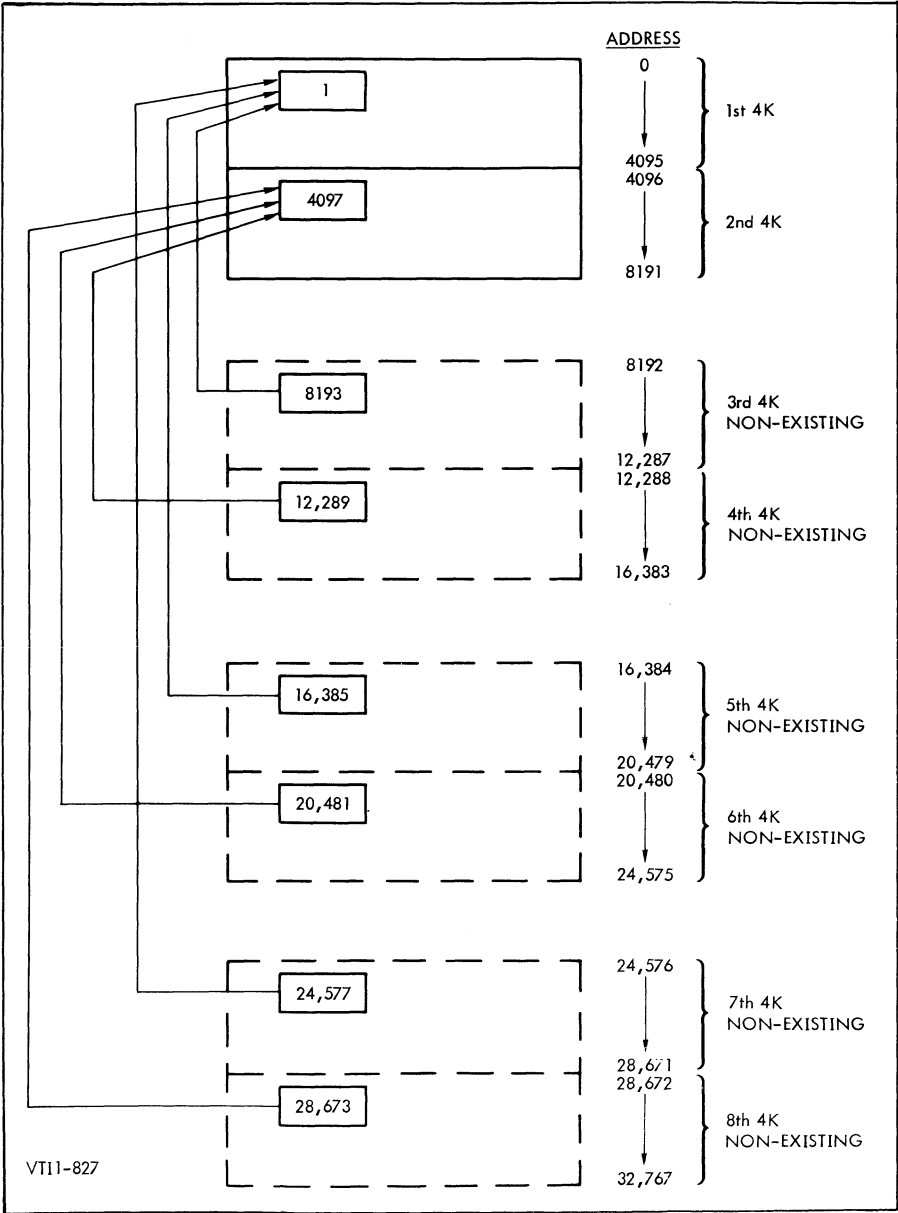


Figure 9-3. Wrap-Around Addressing, 8K Memory

and 16K can be implemented. For 4K wrap-around, the three most significant significant address bits are connected to ground; for 8K wrap-around, the two most significant address bits are grounded; and, for 16K wrap-around, only the most significant address bit is grounded.

An example of 8K wrap-around addressing in a computer with 8K of memory is illustrated in Figure 9-3. In this example, the nonexistent addresses are redirected to their corresponding existing memory locations (1 and 4,097). An example of 16K wrap-around addressing in a computer with 12K of memory is illustrated in Figure 9-4. In this

example, two nonexistent addresses (16,385 and 24,577) are directed back to their corresponding memory locations that do exist (1 and 8,193).

In this particular configuration, however, if the CPU tries to address data to or from nonexistent locations in the eighth 4K-memory increment, the data are not saved because they are directed back to another nonexistent location (fourth 4K-memory increment). In the example of Figure 9-4, the computer must have a 16K memory for all memory addresses to be covered by the wrap-around feature.

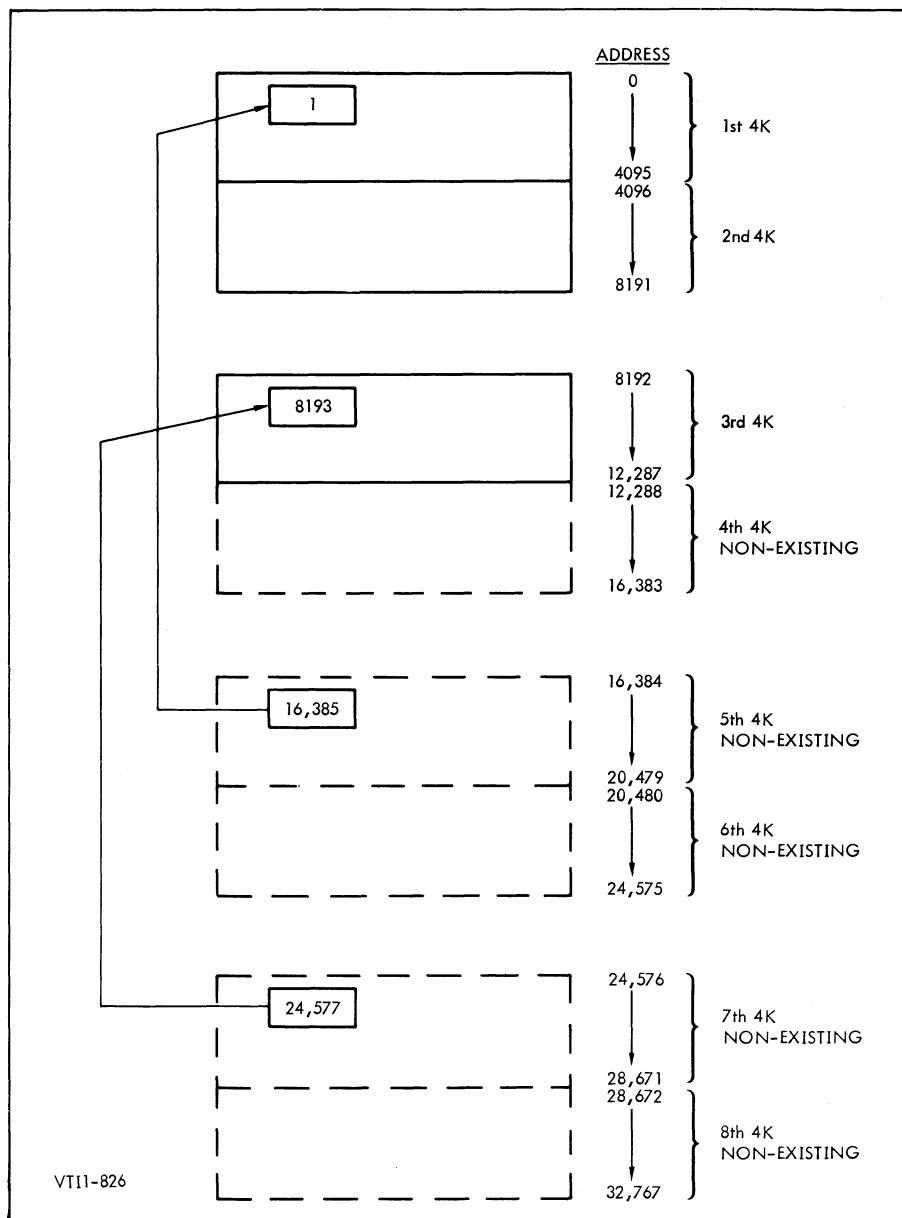


Figure 9-4. Wrap-Around Addressing, 12K Memory

SECTION 10 – MAINFRAME OPTIONS

Five mainframe options are available for the Varian 620/f computer:

1. Special instruction set
2. Memory protection
3. Real time clock
4. Power failure/restart
5. Automatic Bootstrap Loader

Provisions are made within the mainframe enclosure for the installation of all five option cards. These are in addition to the I/O options detailed in Section 12.

Special Instruction Set

This option provides a hardware multiply and divide capability, plus the following two special instructions:

Bit Test (BT)

Skip if Register Equal (SRE)

During multiply, the contents of the B register are multiplied by the contents of a specified memory location. The original contents of the A register are added to the final product. Execution time is 18 microseconds.

During divide, the contents of the A and B registers are divided by the contents of a specified memory location. Execution time is 18 to 25.2 microseconds.

Specifications and instructions for hardware multiply/divide are summarized in Figure 10-1. The BT and SRE instructions are described on page 20-18c.

Memory Protection (MP)

The MP option prevents unauthorized program entry and modification in areas of core memory designated as protected area by programs residing in unprotected areas.

The MP divides core into consecutive 512-word segments of core. Each segment is designated a protected or an unprotected area. The protected/unprotected status of each segment is stored in the MP in four 16-bit mask registers that are loaded by I/O instructions from the CPU. The mask registers can store the status of up to 64 segments, i.e., up to 32K words.

When a program is operating from an unprotected segment, the MP option detects the following operations as errors:

1. Program overflow into a protected area
2. Writing in a protected area
3. Jumping to a protected area
4. Executing an I/O instruction in an unprotected area
5. Executing a halt instruction in an unprotected area

If these operations are attempted, the pro-

gram aborts and jumps to one of eight pre-assigned memory addresses. From these addresses, the program can be directed to subroutines for analysis and correction.

Indirect addressing with the MP option is limited to four levels (Section 19). Further indirect addressing is ignored.

With the PM option, the Varian 620/f becomes a high-reliability, real-time processor that can not be stalled. All unimplemented op codes become non-ops; all instructions are interruptable except EXC, HLT, and the instruction following an XEC. Attempted execution of these instructions is detected as an error.

Specifications for the MP option are given in Figure 10-2.

Real-Time Clock

The RTC provides flexible time orientation that can be used for time-of-day accumulation and as an interval timer. Known periods of time are generated by the RTC for program-sequencing. A free-running counter that can be read under program control is also provided.

The main program is periodically interrupted by the RTC to initiate subroutines which accomplish the desired real-time functions. The RTC can generate two types of interrupts: variable interval and memory overflow. The first interrupt is a time-base signal that increments a specific memory address when recognized by the computer. The second interrupt occurs when the incremented memory address reaches a count of 040001. The frequency of the increment interrupt is adjustable with program-controlled I/O and can range from 3 to

10,000 per second. A predelivery factory adjustment sets the RTC to initiate one increment interrupt every millisecond unless otherwise programmed. An externally generated time base can be used.

Power Failure/Restart

The PF/R provides an orderly shutdown in case of power failure or turnoff and restarts the program when power is restored.

Power input to the computer is indirectly monitored by circuits in the computer power supply. A power failure alarm voltage in the power supply is constantly sensed by the PF/R to determine power status. If the alarm voltage drops (because of power failure or power switch turn-off), the PF/R causes an interrupt. The CPU then executes a user-programmed subroutine that places the contents of volatile registers in memory and halts, after which the CPU and memory are disabled. The power-down service routine cannot be interrupted by lower-priority options or controllers.

When power is restored, the PF/R enables the memory. The CPU executes a user-programmed subroutine that restores the contents of the volatile registers, and the system resumes service of the program in progress at the time of the interrupt.

Automatic Bootstrap Loader

The automatic bootstrap loader (ABL) is a hardware option that automatically loads the 620/f bootstrap program into core memory from an integrated-circuit ROM. This option saves the time and effort involved in manually loading the bootstrap program.

Parameter	Description
Multiply/Divide (M/D)	
Organization	Contains multiply/divide and shift
Multiply Algorithm	$A + (B \cdot R) = A, B$ where A = Initial A register content B = Multiplier (in B register) R = Multiplicand (in memory) A, B = product plus A, in A and B register; most significant half in A, least significant half in B)
Multiplication Capability	Maximum multiplier: 32,767 or -32,768 Maximum multiplicand: 32,767 or -32,768 Maximum product: 1,073,741,824 or -1,073,709,056
Divide Algorithm	$(A, B) / R = B + A$ where A, B = Dividend (in A and B registers; most significant half in A, least significant half in B) R = Divisor (in memory) B = Quotient (in B register) A = Remainder (in A register)
Division Capability	Maximum divisor: 32,767 or -32,768 Maximum dividend: 1,073,741,824 or -1,073,709,056 Maximum quotient: 32,767 or -32,768

Figure 10-1. Hardware Multiply/Divide

Parameter	Description
Organization	Consists of A bus receive, device address decode, function control, mask register, segment address register, instruction address register, protected address detect, error detect, CPU control, interrupt request, and A bus drive sections
Capability	Protects up to sixty-four 512-word memory segments
Timing Sources	18-MHz and 9-MHz clocks from CPU
Prerequisite Options	None
I/O Capability	Six external control and eight data transfer instructions
Priority Assignment	Determined by location in the 620/f priority chain; normally the highest with respect to the DMA interrupt priority assignment
Logic Levels	Positive Logic True: +2.5 to +2.25V dc False: 0.0 to +0.45V dc Negative Logic True: 0.0 to +0.45V dc False: +2.4 to +5.25V dc
Size	One 3-by-15-inch (7.7 x 38.0 cm) wired-socket card
Interconnection	Plugs into 620/f CPU motherboard
Input Power	+5.0V dc at 2.5 amperes
Operational Environment	0 to 50 degrees C; 0 to 90 percent relative humidity without condensation

Figure 10-2. Memory Protection Specifications

Parameter	Description
Priority Assignment	Normally third-highest with respect to DMA/ interrupt priority assignment (exceeded only by MP and PF/R)
I/O Capability	Seven external control and two transfer instructions
Options	Three hardware-selected timing sources for the variable interval interrupt; three hardware-selected timing sources for the free-running counter
Operating Modes	Interval interrupt, memory overflow interrupt, interval accumulation, time-of-day accumulation, and event accumulation
Timing Sources:	
10 kHz	+0.1 percent, square wave; variable interrupt range, 100 microseconds to 409.5 milliseconds (in 100-microsecond increments)
Line Frequency	24V rms, sine wave; variable interval interrupt range (nominal), at 60 Hz 16.7 milliseconds to 68.3 seconds (16.7-millisecond increments) and at 50 Hz 20.0 milliseconds to 81.9 seconds (20.0-millisecond increments)
External (User Supplied)	5.0 microseconds minimum positive duration, 5.0 microseconds minimum negative duration
Free-Running Counter	077777 capacity
Logic Levels:	
I/O	Negative Logic True: -0.5 to +0.5V dc False: +2.4 to +5.0V dc
Internal	Positive Logic True: +2.4 to +5.0V dc False: -0.5 to +0.5V dc
Size	One 3-by-15-inch (7.7 x 38.1 cm) wired-socket card; shares card with PF/R
Interconnection	Plugs into 620/f CPU motherboard

Figure 10-3. Real-Time Clock Specifications (1 of 2)

Parameter	Description
Input Power	+5V dc at 2.0 amperes
Operational Environment	0 to 50 degrees C, 0 to 90 percent relative humidity without condensation

Figure 10-3. Real-Time Clock Specifications (2 of 2)

Parameter	Description
Organization	Consists of a control sequencer, interrupt request logic, and control signal logic
Subroutines	SAVE at location 040 and RESTORE at location 042
Priority Assignment	Normally second-highest with respect to DMA/ interrupt priority assignment (exceeded only by MP)
Power-Down Timing	Disables memory and the CPU less than 2 milliseconds after power loss; up to 300 microseconds allowed for power-down sub-routine
Power-Up Timing	Less than 40 milliseconds after the restoration of full power
Logic Levels: I/O	Negative Logic True: 0.0 to +0.5V dc False: +2.8 to +3.6V dc
CPU and Memory	Positive Logic True: +2.4 to +5.5V dc False: 0.0 to +0.5V dc
Size	Occupies approximately 15 IC sockets on a 3-by-15-inch (7.7 x 38.1 cm) circuit card; shares card with RTC
Interconnection	Plugs into 620/f CPU motherboard
Input Power	+5V dc at 0.5 ampere
Operational Environment	0 to 50 degrees C, 0 to 90 percent relative humidity without condensation

Figure 10-4. Power Failure/Restart Specifications

Parameter	Description
Organization	Consists of an IC ROM address counter, control logic, and BOOTSTRAP switch on the control panel
ROM Program	Standard ABL option provided with the ROM programmed for the high-speed paper tape reader or TTY. As a special option, the standard ROM can be replaced with one specially programmed for any other I/O device if the bootstrap program does not exceed thirty-two 16-bit words
Data Transfer	16-bit transfers from the ROM to the 620/f memory via the DMA channel; hardware insertion of instructions in the P, I, and X registers
Options	Specially programmed ROMs for various I/O devices, card reader, drum, disc, magnetic tape units, etc.
Logic Levels	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to +0.5V dc
Size	Occupies approximately 25 IC sockets on a 3-by-15-inch (7.7 x 38.1 cm) circuit card; shares card with first TTY controller
Interconnection	Plugs into 620/f CPU motherboard
Input Power	+5V dc at 900 mA
Operational Environment	0 to 50 degrees C, 0 to 90 percent relative humidity without condensation

Figure 10-5. Automatic Bootstrap Loader Specifications

SECTION 11 – INPUT/OUTPUT SYSTEM

The Varian 620/f input/output system provides a powerful and flexible interface between the computer and the peripheral devices that are connected to it.

The basic I/O system has three principal parts: The timing-and-control logic within the CPU, a series of peripheral-device controllers, and a party-line, time-shared I/O bus that interconnects the CPU and the controllers.

The CPU portion of the system has been described in Section 8. Details on the controllers for specific peripheral devices, are given in Section 13. The present section concentrates on the role played by the I/O bus and the interaction between the I/O bus and the peripheral controllers.

Basic I/O Options

The capabilities of the basic I/O system can be greatly expanded by the addition of either or both of two I/O options:

The Priority Interrupt Module (PIM), which allows any peripheral controller in the system to initiate its own interrupt subroutine.

The Buffer Interface Controller (BIC), which allows high-data-rate peripherals to transfer data to or from memory at rates up to 202,000 words per second.

The I/O bus, backplane wiring, and CPU logic of the standard Varian 620/f are fully implemented for both these options. Consequently, the following description of the I/O system assumes that both options are included. Details on the options themselves are given in Section 12.

PMA I/O Option

For very high-speed data transfers (over a million words per second) the Varian 620/f can be implemented with a Priority Memory Access (PMA) channel separate from the basic I/O system. The modules and system interconnections required for the PMA option are also described in Section 12.

I/O System Functions and Organization

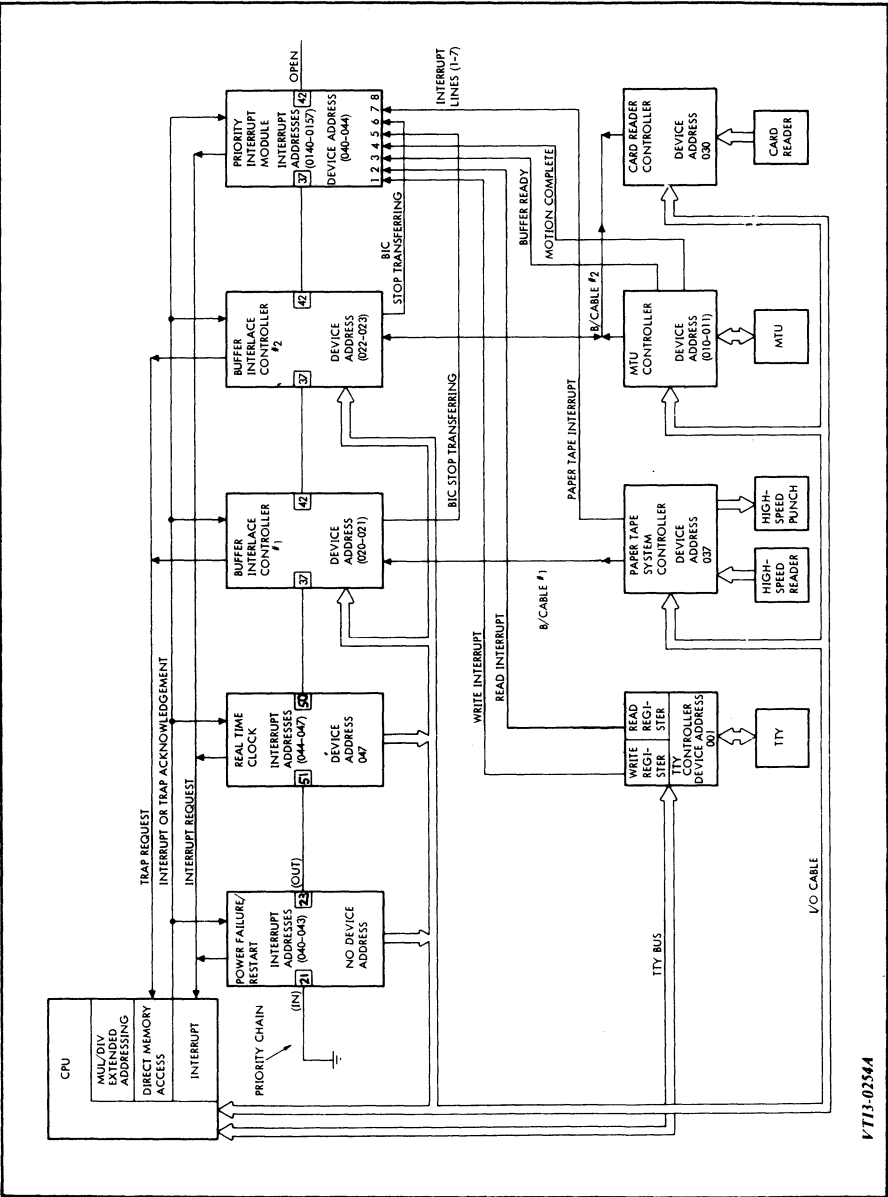
The I/O system is organized as shown in Figure 11-1. All of the peripheral controllers and I/O options, other than the first teletype controller, are connected to the I/O bus. The I/O-bus connections within the mainframe chassis and the expansion chassis take the form of backplane wiring; chassis-to-chassis I/O-bus connections are via an I/O cable that interconnects the backplanes of each chassis.

The priority chain, B cable, interrupt-request, trap-request, and acknowledgement lines shown in Figure 11-1 are also included in the I/O bus, but are shown separately for the sake of clarity. Only the interrupt lines into the PIM are separate entities, installed as needed.

I/O Bus Structure

The I/O bus that interconnects the peripheral controllers, I/O options and CPU has four structural elements:

- a. A set of 16 bi-directional lines (EB00-I through EB15-I) called the E bus. These lines serve a variety of data-transfer purposes, including the transfer of operands, peripheral-device addresses, memory-location addresses, peripheral-control codes, and peripheral-sense codes.



VT13-02544

Figure II-1. Typical I/O System Configuration

- b. A set of 10 individual control lines that either (1) signal the type of information present on the E bus, (2) provide timing and synchronization between the CPU and the controllers, or (3) initiate a specific action or sequence on the part of the CPU or controllers. The ten control lines are identified and described in Figure 11-2. The relationship between the control lines, the E-bus lines, and the operation in process is summarized in Figure 11-3.
- c. A set of four unconnected lines (PR1X-I through PR4X-I) that are used to make up the interrupt-priority chain.
- d. A set of seven "B cable" control lines for communication between each BIC and its peripheral controllers.

Drivers and receivers are contained in the CPU and controllers for signalling over the E-Bus and control lines. Figures 11-4 and 11-5 are schematic drawings of these circuit elements. Standard I/O drivers are capable of servicing up to 10 peripheral controllers. When more than 10 controllers are included in the system, an I/O Buffer Card must be added to the network.

The E-bus lines are bi-directional; the control lines are uni-directional, directing signals either from or to the CPU. Circuit elements for all three types of lines are shown in Figures 11-6, 11-7, and 11-8. The lines are logically true at 0 Vdc and logically false at +3 Vdc. The termination shoes shown in the drawings are mounted on a single card that is inserted into an I/O-bus slot following the last controller in the system.

I/O Modes of Operation

There are three modes of operation for

the basic I/O system (see Section 12 for the PMA mode):

Programmed I/O operations are a result of I/O instructions written into the main computer program or into subroutines called by the main program.

Interrupt-initiated I/O operations are identical to the equivalent programmed I/O operations except that they are initiated by an interrupt request generated by a peripheral controller or I/O option. The interrupt branches the program to a memory-address location that generally contains a jump-and-mark instruction. This, in turn, directs the program to an I/O subroutine. In order for the main program to resume, the subroutine must end with an instruction that returns the program to the location stored at the mark address.

Cycle-stealing I/O is an optional mode that is implemented by the addition of a Buffer Interlace Controller (BIC). It combines features of both the programmed I/O and the interrupt-initiated I/O modes. The trap requests (see Figure 11-2) are the equivalent of the interrupt requests except that they inhibit the progress of the main program for only the 3.6 microseconds required to transfer a word of data directly between memory and the peripheral device. The direct-memory-access (DMA) feature of the cycle-stealing mode allow data words to be transferred at rates up to 276,000 words per second, as compared to a typical 30,000 words per second for programmed I/O.

An interrupt-initiated I/O transfer branches the program to a subroutine that must include instructions returning the CPU to the main program. A cycle-stealing I/O transfer, on the other hand, only inhibits the main program while a single word of data is transferred. The operational registers are unaltered, and at the end of a 2.1 microsecond time interval, the main program automatically resumes.

Meaning •	Mnemonic	Function
Control code and address on E bus	FRYX-I	FRYX-I is generated by the computer to indicate that the computer has placed a device address and a control code on the E bus. Each peripheral controller examines the device address, and, upon the true-to-false transition of FRYX-I, the addressed device responds to the control code.
Sense code and address on E bus	SERX-I	SERX-I is a controller response to a Sense (SEN) instruction. During the execution of the Sense (SEN) instruction, the computer places a function code and a device address on the E bus. The addressed controller is instructed to indicate the status of the specific device condition identified by the function code. If the specified condition is true, the controller responds by setting the SERX-I line true. If the condition is false, SERX-I is left false.
Data on E bus	DRYX-I	DRYX-I is generated by the computer. During an output data transfer, DRYX-I indicates that the computer has placed data on the E bus and that the peripheral device previously addressed should strobe the data into its input buffer. During an input data transfer, DRYX-I indicates that the computer has accepted the data placed on the E bus by the peripheral device, and that, following the true-to-false transition of DRYX-I, the device should remove the data.
Initialize all controllers	SYRT-I	SYRT-I is used to initialize each peripheral device controller connected to the I/O bus. SYRT-I becomes true when the SYSTEM RESET switch on the control console is pressed.
Interrupt synchronization clock	IUCX-I	IUCX-I is a 1.1-MHz clock from the computer that is disabled by IUAX-I. When IUAX-I is false, the clock is present on the IUCX-I line. When IUAX-I is true, IUCX-I is held true. The true-to-false transition of IUCX-I is used to set the request flip-flops (IURX-I, TPOX-I, TPIX-I) in respective controllers.
Interrupt request	IURX-I	By setting IURX-I true, the interrupting device controller requests the computer to execute an instruction. The address of this instruction is placed on the E bus by the interrupting device upon receipt of IUAX-I from the computer.

Figure 11-2. I/O Bus Control Lines (1 of 2)

Meaning	Mnemonic	Function
Interrupt acknowledge	IUAX-I	IUAX-I is set true by the computer to acknowledge the receipt of an interrupt, a trap in, or a trap out. The interrupting or trapping device controller can communicate an address to the computer and can receive data from or send data to the computer only when this control signal is true. IUAX-I is also used to inhibit device address decoding in every device controller during the address phase of an interrupt or trap operation. This prevents the controllers from interpreting the lower-order bits of a memory address as a device address.
Jump-and-mark interrupt	IUJX-I	IUJX-I is generated by the computer to inhibit all interrupts that occur after a jump-and-mark instruction when that instruction is the result of an interrupt request. The signal becomes true 2.7 microseconds from the false-to-true transition of IUAX-I and remains true for 450 nanoseconds.
Trap input request	TPIX-I	By setting TPIX-I true, the optional Buffered Interface Controller requests the computer to input one word of data to memory. The address of this word is placed on the E bus by the controller upon receipt of IUAX-I from the computer.
Trap output request	TPOX-I	By setting TPOX-I true, the optional Buffered Interface Controller requests the computer to output one word of data from memory. The address of this word is placed on the E bus by the controller upon receipt of IUAX-I from the computer.

Figure 11-2. I/O Bus Control Lines (2 of 2)

Programmed I/O Operations

Four types of I/O operations can be performed under program control:

External Control. An external control code

is transmitted under program control from the computer to a peripheral controller.

Program Sense. The status of a selected peripheral controller sense line is interrogated by the computer under program control.

Operation					
Control Lines	External Control	Sense	Data Transfer (Single Word I/O)	Trap Sequence (Buffer Interface Control)	Interrupt Sequence
EB00-I	FRYX-I* (Phase 1)	FRYX-I* SERX-I* (Phase 1)	FRYX-I* (Phase 1)	TPOX-I or TPIX-I	IURX-I
EB05-I to EB06-I	Device Address	Device Address	Device Address	---IUAX-I, FRYX-I (Phase 1)	IUAX-I (Phase 1)
EB08-I to EB09-I	Function Code	Function Code	Not Used	---IUAX-I, FRYX-I (Phase 1)	IURX-I (Phase 1)
EB10-I	Not Used	Not Used	Not Used	---	---
EB11-I	External Control Command			---	---
EB12-I		Sense Command		---	---
EB13-I			Data Transfer In	---	---
EB14-I			Data Transfer Out	---	---
EB15-I	Extended External Control Command			---	---

*IUAX Interlock - used in the address decoding term.

Figure 11-3. Relationship Between E-Bus and I/O Control Lines

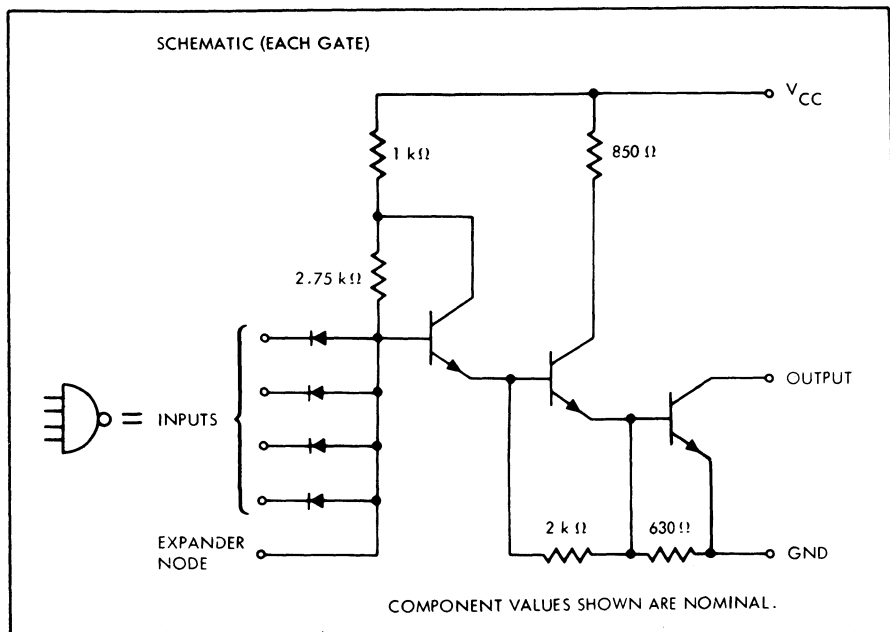


Figure 11-4. I/O Bus Driver

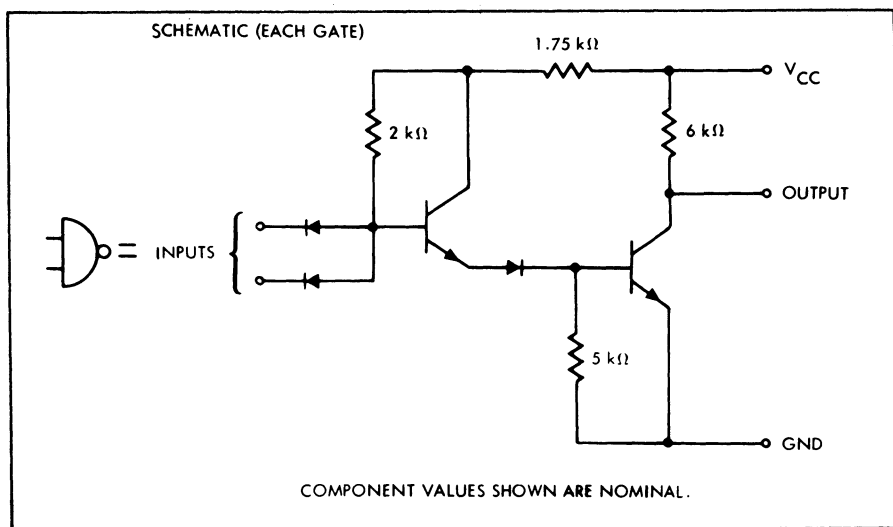
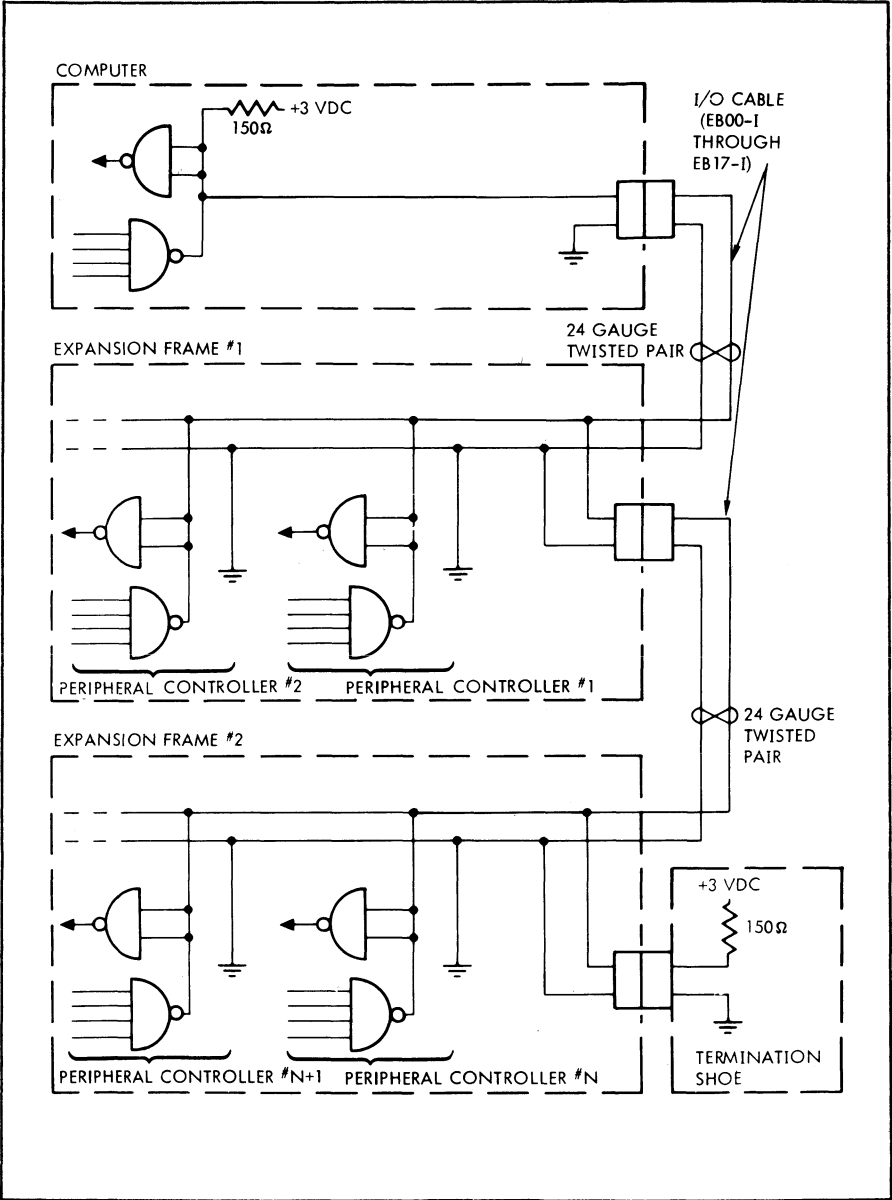
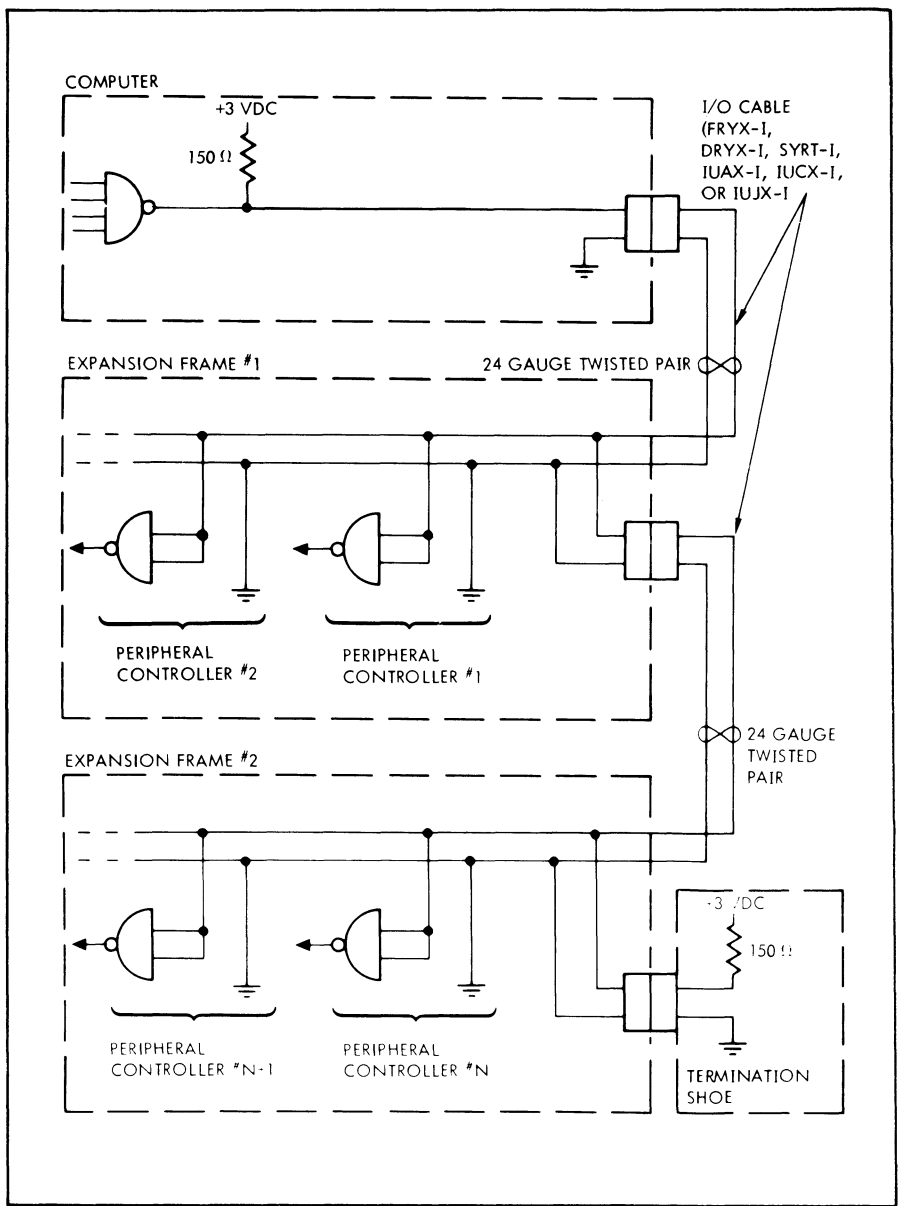


Figure 11-5. I/O Bus Receiver



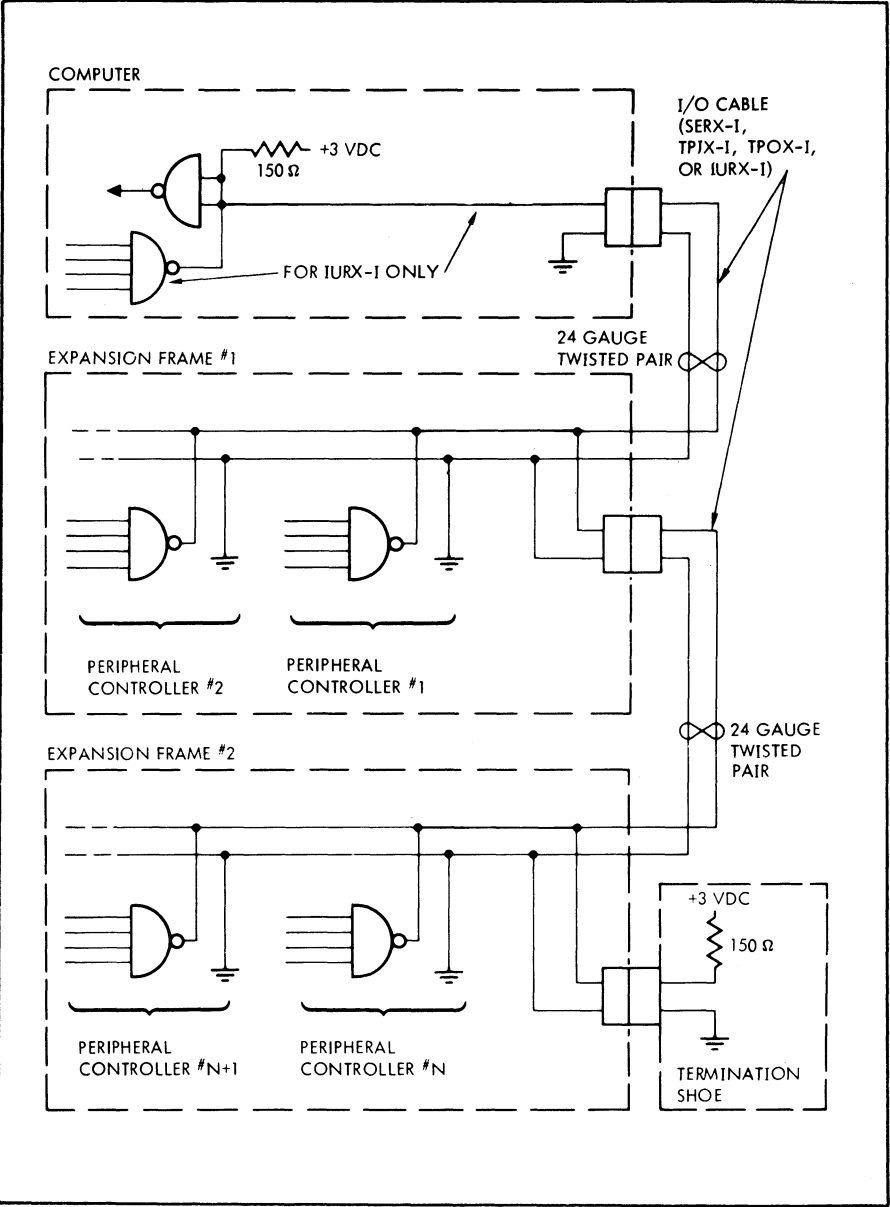
VT11-1239

Figure 11-6. Typical E-Bus Line



VT11-1240

Figure 11-7. Typical Control From The Computer



VT11-1243

Figure 11-8. Typical Control Signal To The Computer

Single-Word Input Transfer. A single word of data is transferred under program control from a peripheral controller to the A register, B register, or any location in memory.

Single-Word Output Transfer. A single word of data is transferred under program control to a peripheral controller from the A register, B register, or any location in memory.

External Control

The external control (EXC) instruction is used to initiate a specific mode of operation in a peripheral device. An example is the use of an EXC command to cause a magnetic tape transport to advance the tape one record. The EXC instruction word format is shown below, where YY contains the device address and X contains the function code.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

10	0	X	YY
----	---	---	----

The EXC instruction causes the function codes and device address portions of the instruction word to be placed on the E bus. The EXC I/O timing is shown in Figure 11-9. Signal lines EB00-I through EB05-I indicate the device address; EB06-I through EB08-I indicate the function code. EB11-I is held true, indicating that an EXC function is being performed (as noted below, EB11-15 is used when additional function codes are required.)

The device controller decodes the binary function code, and, following the true-to-false transition of FRYX-I, initiates the specified mode of operation in the addressed device. During the execution of EXC, no data are exchanged between the computer and the device controller, and no

response signal is expected from the controller.

When additional function codes are required, the extended EXC instruction (instruction code 104) can be used. This instruction is identical to the normal EXC instruction (instruction code 100) in timing and function, but it is identified by EB15-I instead of EB11-I.

Figure 11-10 shows typical implementation of the logic required in a peripheral controller to receive, decode, and perform an EXC instruction.

Program Sense

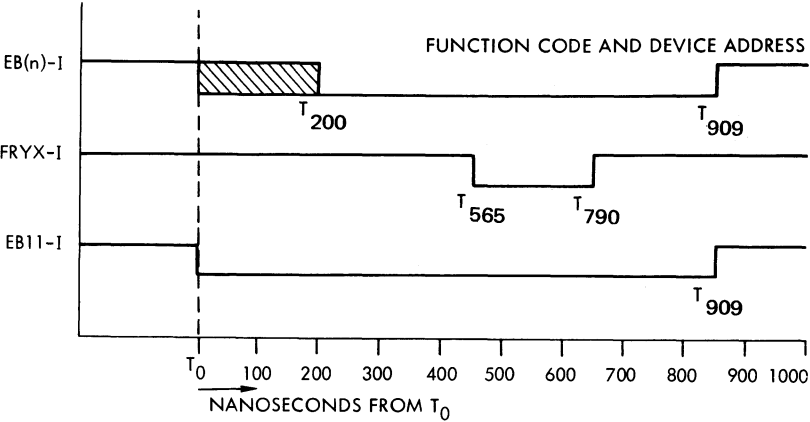
The program Sense (SEN) instruction is used to test the status of a specific device condition, and, if a true condition is detected, a program jump is made. If a false condition is detected, the next instruction in sequence is executed. An example of SEN instruction usage is a test to determine if a magnetic tape transport is rewinding.

The SEN instruction format is shown below, where YY contains the device address and X contains the function code, which defines the specific condition to be tested.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	1	X	YY
I	Jump Address		

The SEN instruction causes the function code and device address portions of the instruction word to be placed on the E bus. The SEN I/O timing is shown in Figure 11-11. Signal lines EB00-I through EB05-I indicate the device address. EB12-I is held true to indicate that a SEN instruction is in progress.



T_0 is the start of the execute phase of the external control instruction.

Logic levels: true = 0 VDC
false = +3 VDC

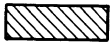
 = time when signal is settling.

Figure 11-9. External Control Timing

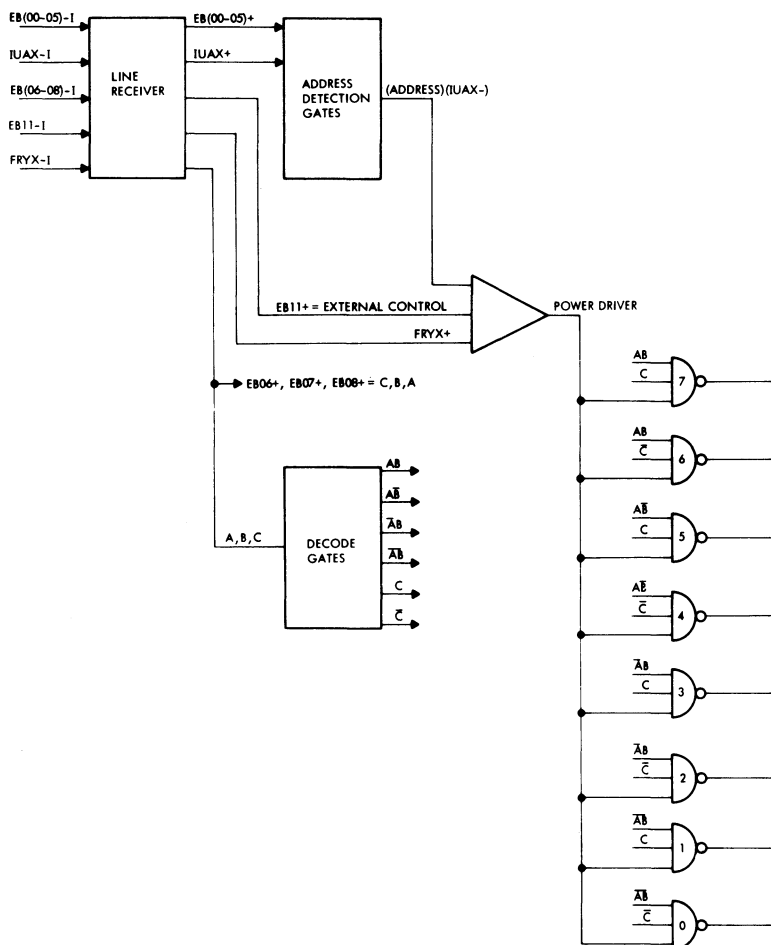
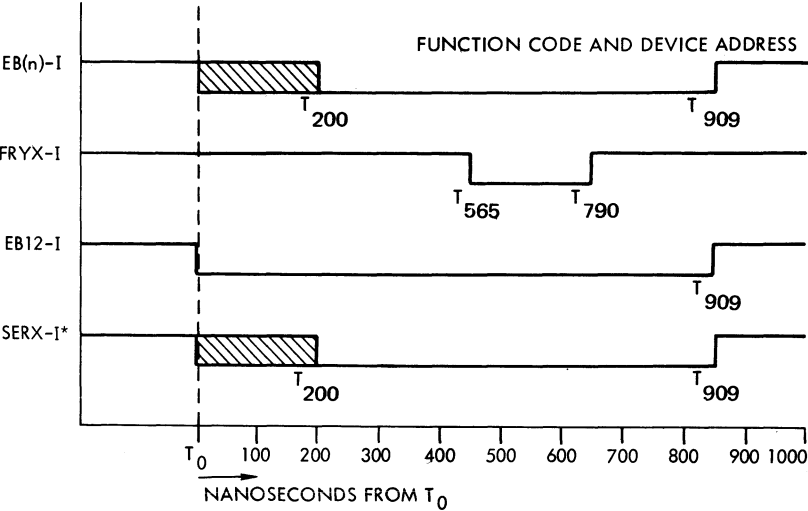


Figure 11-10. Typical Peripheral Controller Logic: EXC Command



T₀ is the start of the execute phase of the sense instruction.

Logic levels: true = 0 vdc,
false = +3 vdc.

 = time when signal is settling.

*SERX-1 is normally on at T₂₀₀; it must be on by T₃₄₀

Figure 11-11. Sense Response Timing

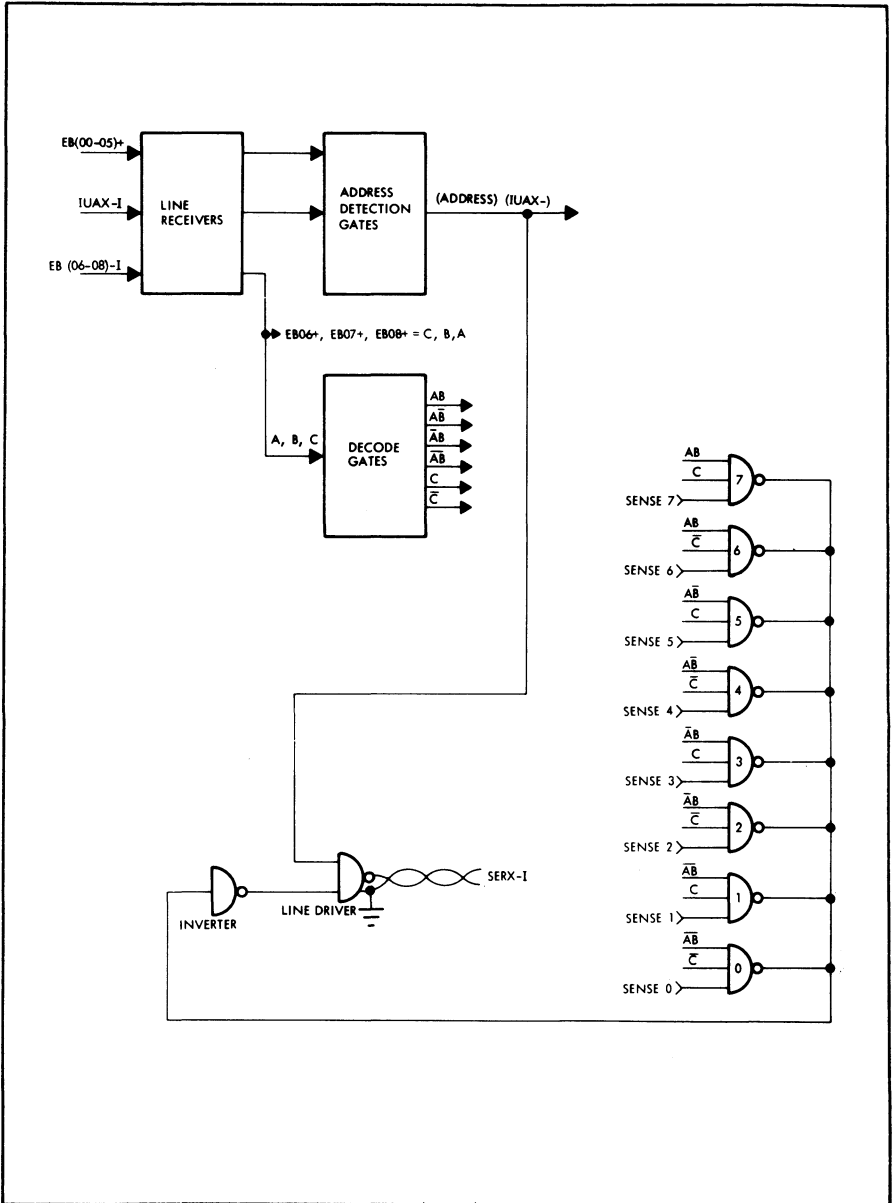


Figure 11-12. Typical Peripheral Controller Logic: SEN Command

Figure 11-12 shows typical implementation of the logic required in a peripheral controller to receive, decode, and respond to a SEN instruction. Note that the device address (usually ANDed with IUAX-I) can be used directly to enable the sense line response (SERX-I). EB12-I need not be used to enable SERX-I, since the computer samples the SERX-I line only when a SEN instruction is executed.

Input Transfer

Seven instructions provide a single-word input transfer:

- Input to A register (INA)*
- Input to B register (INB)*
- Input to A and B (INAB)
- Input to memory (IME)
- Clear and input to A register (CIA)
- Clear and input to B register (CIB)
- Clear and input to A and B (CIAB)

The instruction word format for INA, INB, INAB, CIA, CIB and CIAB is shown below, where YY contains the device address and X defines each individual command.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	X	YY
----	---	---	----

The instruction word format for IME includes a second word that specifies the input data word destination.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	0	YY
0	Data Address		

*Inclusive OR of the data lines and the specified register contents.

The execution of any one of the seven single-word input transfer instruction produces the same sequence of operations on the I/O bus. Consequently, in responding to the bus signals, a peripheral device controller makes no distinction between the input transfer instructions.

The execution of an input transfer instruction is a two-phase operation. The first phase selects the peripheral device controller that will participate in the second-phase data transfer. The transfer timing is shown in Figure 11-13. The first phase is initiated by the computer, which places the device address on E bus lines EB00-I through EB05-I. EB13-I is held true during this phase to indicate that an input transfer instruction is in progress. Since the E bus is time-shared, a flip-flop in the peripheral controller for the selected device is set to indicate that the controller was selected and that data are to be transferred to the computer. This flip-flop, Data Transfer In (DTIX+), is set at the true-to-false transition (trailing edge) of FRYX-I (if the controller controls more than one device, an additional flip-flop for each device is required to identify the selected device. As the computer removes the device address and control code information, the selected controller uses DTIX+ to enable the input data onto the E bus. The controller must enable data from the selected device onto the bus no later than 710 nanoseconds after the trailing edge of FRYX-I to strobe the input data. The controller uses the trailing edge of DRYX-I to reset DTIX+, thus removing the input data from the E bus.

When the computer is transferring I/O data under program control, the transfers must be synchronized with the communicating peripheral controller. This synchronization is accomplished by sampling the state of the controller for a ready condition by issuing

SEN instruction prior to the data transfer instruction.

Figure 11-14 shows typical implementation of the logic required in a peripheral control to perform an input data transfer.

Output Transfer

There are four single-word output instructions:

- output from A register (OAR)
- Output from B register (OBR)
- Output the inclusive OR of A and B (OAB)
- Output from memory (OME)

The instruction word format for OAR, OBR, and OAB is shown below, where YY contains the device address and X distinguishes between the two instructions.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10				3		X		YY							

This instruction word format for OME (shown below) includes a second word that specifies the output data-word source location.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10				3		0		YY							
0	Data Address														

The sequence of signals placed on the I/O bus is the same for each of the three output instructions, and the participating peripheral device controller makes no distinction between the output commands.

The execution of an output command is a two-phase operation similar to that described for an input command. The first

phase selects the peripheral device which will participate in the second-phase data transfer. The transfer timing is shown in Figure 11-15. The first phase is initiated by the computer, which places the device address on E-bus lines EB00-I through EB05-I. EB14-I is held true during this phase to indicate that an output transfer command is being executed. A flip-flop, data transfer out (DIOX+), in the controller for the selected device is set at the trailing edge of FRYX-I (if the controller controls more than one device, an additional flip-flop for each device is required to identify the device selected). Following FRYX-I, the computer removes the device address and control code, and places the output data on the E-bus lines. DIOX+ is used by the controller to gate the contents of the E bus into an input buffer at the trailing edge of DRYX-I. The trailing edge of DRYX-I is also used to reset DIOX+.

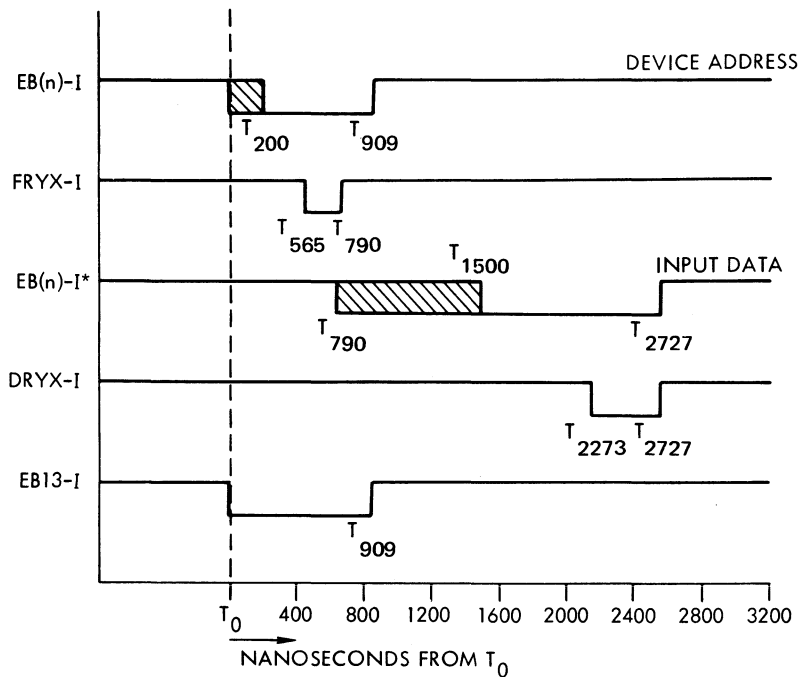
When the computer is transferring I/O data under program control, the transfers must be synchronized with the communicating peripheral controller. This synchronization is accomplished by sampling the state of the controller for a ready condition by issuing a sense command prior to the data transfer command.

Figure 11-16 shows typical implementation of the logic required in a peripheral controller to perform an output data transfer.

Device Address Codes

Device address codes have been assigned to the standard peripheral devices used in Varian 620/f systems.

All device address codes are in the range 00 to 77. Each peripheral device belongs to a class, according to its function. Each class is assigned a block of codes, and specific code



T_0 is the start of the execute phase of the data transfer in instruction.

Logic levels: true = 0 vdc,
false = +3 vdc.

 = time when signal is settling.

* EB(n)-I (input data) must be off by T_{2727} .

Figure 11-13. Data-Transfer-In Timing

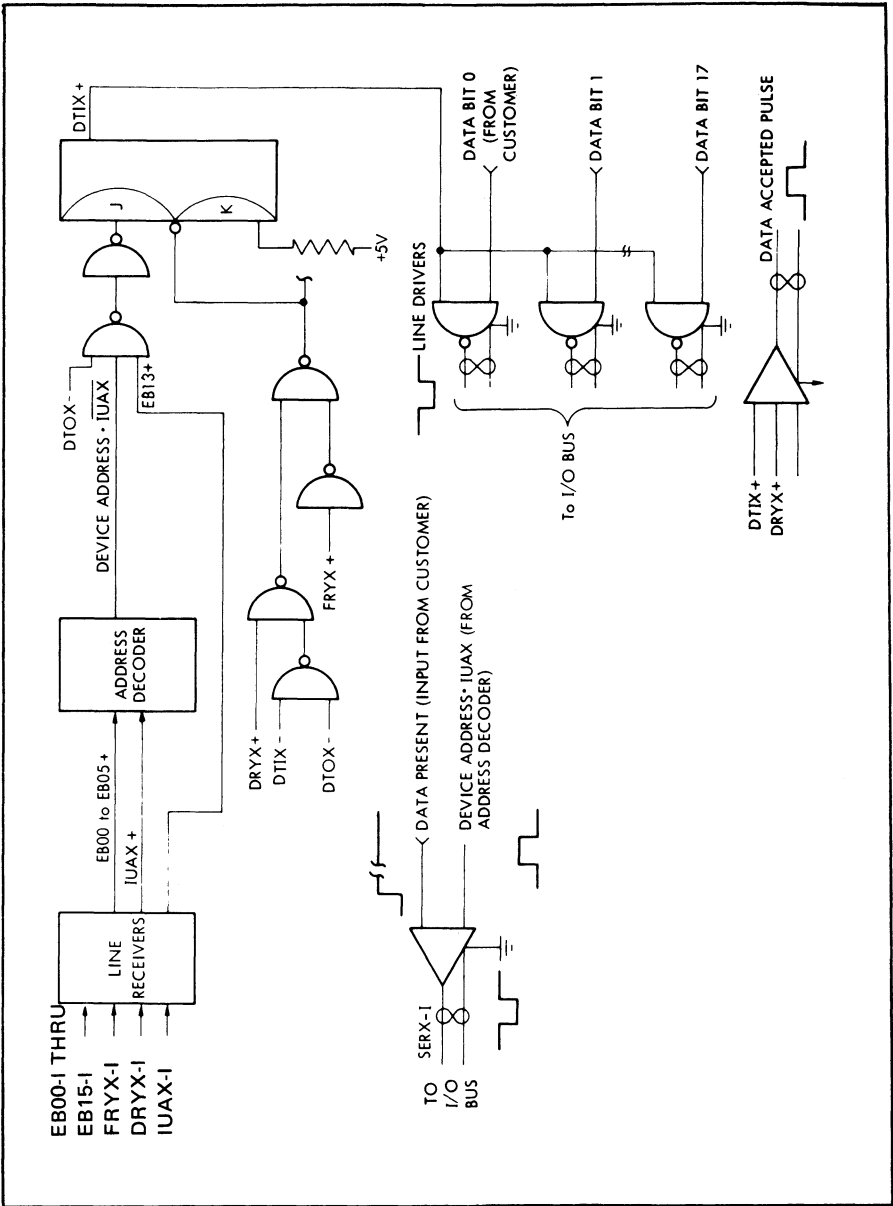


Figure 11-14. Typical Peripheral Controller Logic: Input Data Transfer

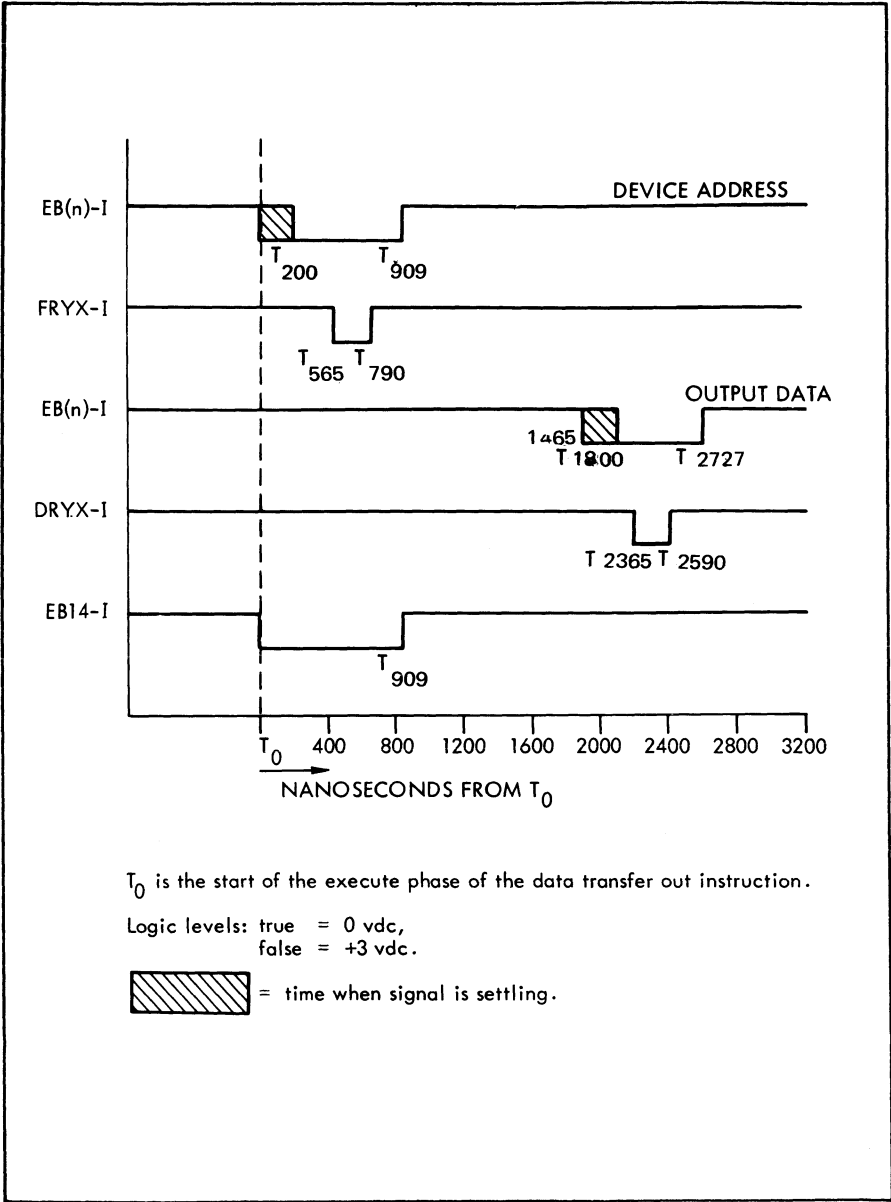


Figure 11-15. Data-Transfer-Out Timing

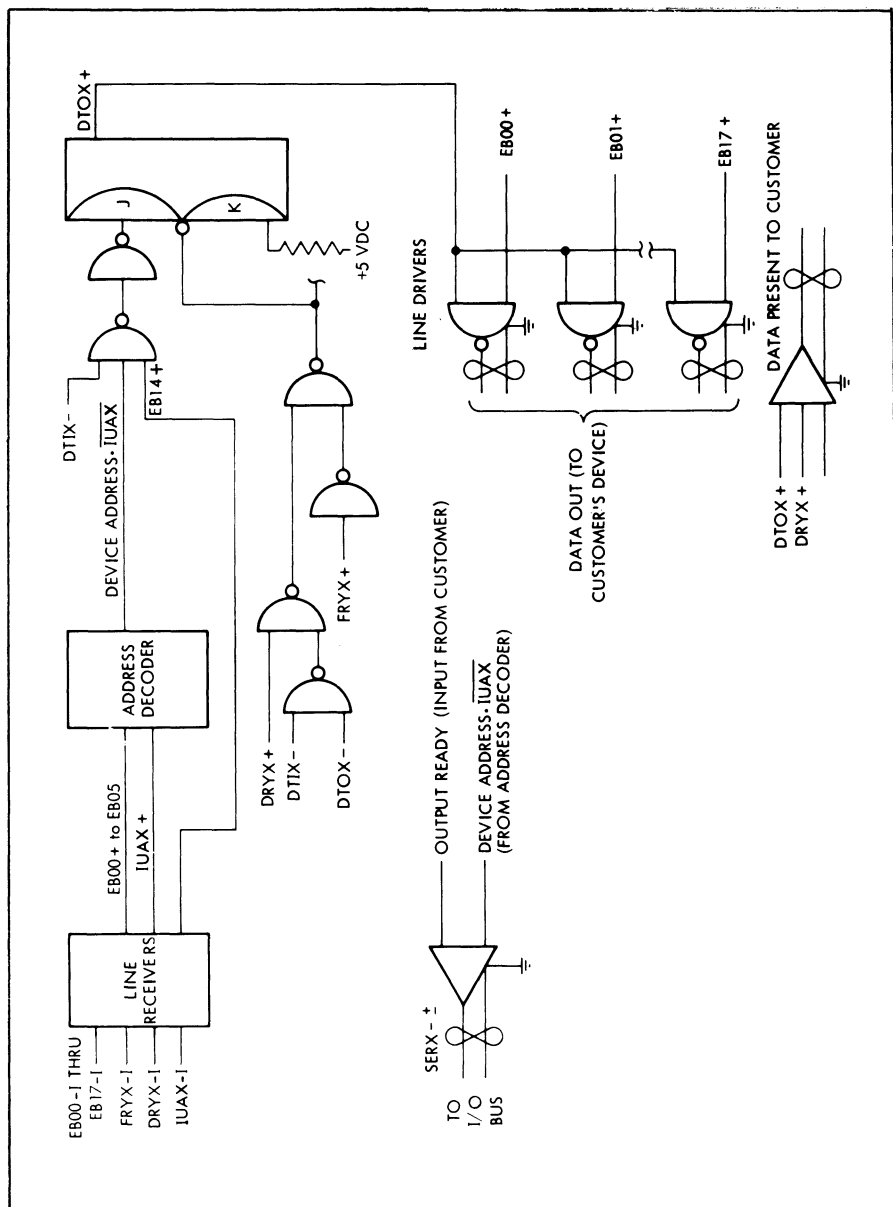


Figure 11-16. Typical Peripheral Controller Logic: Output Data Transfer

interrupt structure

assignments are given to devices in that class.

The device address codes are listed in Figure 20-1, Section 20.

Interrupt Structure

Three of the control lines in the I/O bus (Figure 11-2) are capable of interrupting or inhibiting the program being processed by the CPU. The three lines are IURX-I (Interrupt Request), TPIX-I (Trap-In Request), and TPOX-I (Trap-Out Request).

The three are similar in that (1) the computer responds to all three by an IUAX-I (Interrupt Acknowledge) signal, (2) in each case, a memory location address must be placed on the E bus for the computer to process the request, and (3) controllers generating the three types of requests are all interconnected on the same priority chain.

They differ in that IURX-I interrupt branches the program to a subroutine specified by the interrupt address, whereas the two trap requests, TPIX-I and TPOX-I, direct the computer to transfer data to or from the memory address. The data to be transferred is placed on the E bus immediately after the memory address has been transmitted.

Another difference is that subroutines specified by an IURX-I interrupt must contain instructions returning the CPU to its original program, whereas the CPU automatically returns to its program after transferring a single word of data in response to a TPIX-I or TPOX-I trap request.

The IURX-I interrupt can be generated, in theory, by any controller connected to the I/O bus. In practice, however, the

capability is limited to those units that can also generate the required interrupt address location.

Standard Varian 620/f peripheral controllers do not have this capability; if the IURX-I interrupt is generated by one of these controllers, the computer interprets the lack of an address as an octal 000 and branches the program to the first memory location.

The Priority Interrupt Module (PIM) option, described in detail in Section 12, provides an addressing capability for the peripheral controllers. The peripheral controller directs an interrupt request to the PIM, which in turn generates an IURX-I interrupt and the appropriate interrupt address (hardwired in the PIM circuitry) for that peripheral. Up to eight peripheral controllers can be connected to a single PIM. The PIM contains priority logic to give preference to higher-priority devices in case two peripheral controllers request an interrupt at the same time.

Three of the mainframe options are also part of the interrupt structure. The real-time-clock, power-fail/restart, and memory-protect options (see Section 10) are on the priority chain, can generate a memory address, and use the IURX-I interrupt to branch the CPU to the subroutine specified by the address on the E bus.

Interrupt address locations have been assigned for the mainframe options and the PIM controllers. These are listed in Figure 11-17. Two memory locations are reserved for each interrupt to accommodate double-word instructions, such as the jump-and-mark group.

Interrupt-Initiated I/O

An interrupt-initiated I/O transfer starts

Address (octal)	Peripheral Device
040,041	Power failure interrupt
042,043	Power restart interrupt
044,045	RTC interval interrupt
046,047	RTC overflow interrupt
120,121	MP write error interrupt
124,125	MP jump error interrupt
130,131	I/O error interrupt
134,135	MP overflow error interrupt
100-117	PIM, first module
140-177	PIM, remaining modules

Figure 11-17. Assigned Interrupt Device Addresses

with an interrupt request directed to a PIM controller by a peripheral-device controller. The PIM, in turn, initiates the interrupt sequence that ends with the CPU branching, in most cases, to a subroutine that services the peripheral-device controller making the initial request.

The interrupt sequence is comprised of two phases: request and address, as shown in the interrupt timing sequence, Figure 11-18.

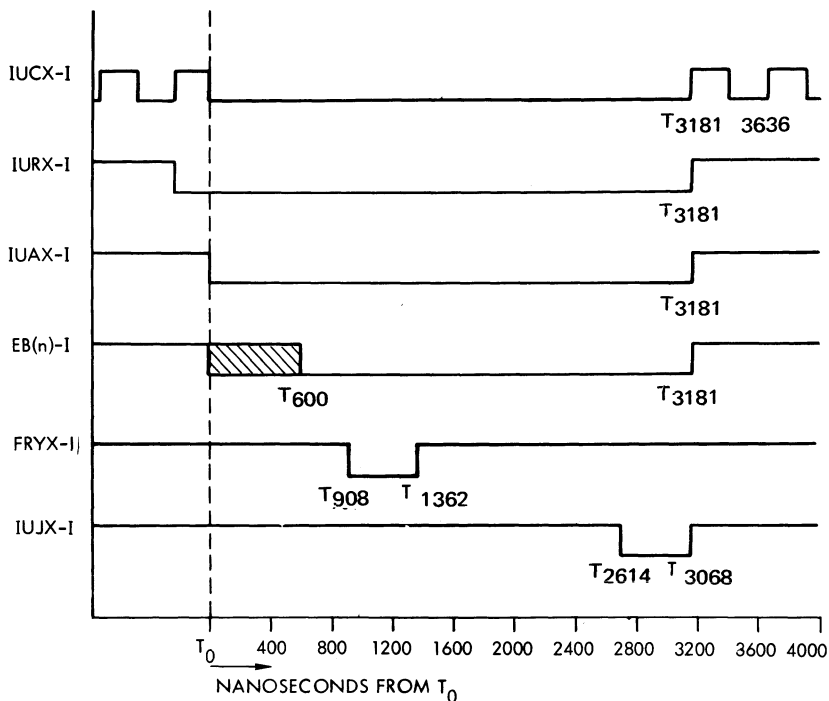
The PIM controller having the highest priority generates an interrupt request (sets URX-I true) on the trailing edge of the interrupt clock (IUCX-I). The controller then waits for an interrupt acknowledge (IUAX-I) from the computer.

When the computer has recognized the

request, it responds by setting IUAX-I true, thus holding IUCX-I true. Upon receipt of IUAX-I from the computer, the requesting device controller places on the E bus the address containing the instruction to be executed.

The instruction at this address may be any instruction except an I/O instruction. In the case of a two-word instruction, the computer ORs a binary one into bit position zero of the second word, making this address an odd number. Because of this, the address placed on the E bus by the controller must be an even number.

If the instruction is a jump-and-mark instruction, the controller receives an interrupt-jump signal (IUJX-I) from the computer. IUJX-I resets the master enable in the PIM controller of every device capable of making a request, thereby inhibiting



T_0 is the start of the timing and interrupt sequence.

Logic levels: True = 0 vdc,
False = +3 vdc.

 = time when signal is settling.

IURX-I is normally off at T_{3181} ; it must be off by T_{3300} .

EB(n)-I is normally on at T_0 ; it must be on by T_{600} . It is normally off at T_{3181} ; it must be off by T_{3300} .

IUJX-I is present for a jump and mark instruction.

Figure 11-18. Interrupt Timing

further interrupts from these PIM controllers during the subsequent subroutine.

Each PIM controller must be re-enabled by the computer program at the end of the subroutine. The address phase ends when IUAX-I becomes false. All signals must be removed from the bus at this time.

Interrupt-initiated I/O transfers (as well as any other type of interrupt) are inhibited under any one of the following conditions:

1. During any two word instruction.
 - *2. During any I/O instruction.
 - *3. Until at least one non-I/O instruction has occurred after an I/O instruction.
 4. During any shift or rotate instruction.
 5. During any shift which occurs as part of a hardware multiply or divide operation.
 6. Immediately following a shift operation during which a trap has occurred.
 - *7. During a manual step.
 - *8. When in step, i.e. when halted.
 - *9. When a manual or programmed halt is in the process of occurring.
 - *10. Until the run mode has been fully entered.
- *Also applies to trap requests

Cycle-Stealing I/O

The cycle-stealing, direct-memory access (DMA) I/O mode is implemented by the addition of one or more (up to 4) Buffered Interface Module options.

The operation of the BIC option is detailed in Section 12. Its two principal functions are to generate the TPIX-I and TPOX-I trap requests and to generate the memory addresses where the transferred data is to be stored or retrieved. The data flow is over the E bus, directly between memory and the peripheral controller, not through the BIC.

A cycle-stealing I/O transfer can be initiated

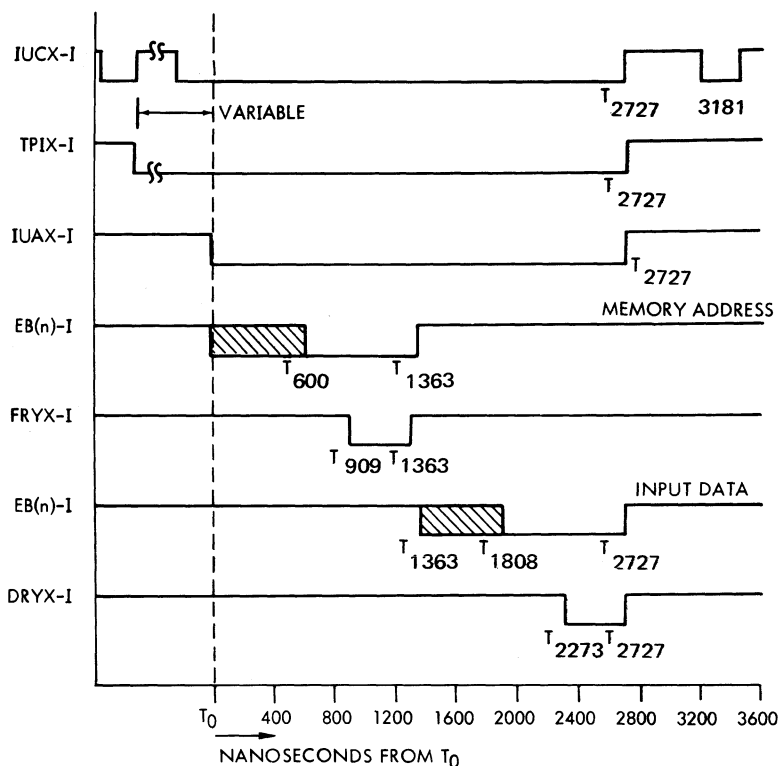
by either computer program or by the generation of a trap request by a peripheral controller. In either case, a subroutine is entered that establishes the initial and final address locations for the transfer, identifies the peripheral controller, and initializes both the BIC and the peripheral controller for the cycle-stealing transfer.

The BIC senses when the peripheral-controller buffer is ready to transmit or accept data, then initiates a TPIX-I or TPOX-I trap request on the trailing edge of the interrupt clock (IUCX-I). When the interrupt acknowledge (IUAX-I) has been received from the computer, it places the first memory address on the E bus and increments the initial address register by one. This completes the first phase of the cycle-stealing I/O transfer.

The second phase is the transfer of data directly to or from memory on the E bus. The memory location accessed is that of the memory address on the E bus during the first phase.

The CPU operational registers are not affected during this transfer. The computer program is simply inhibited during the 3.6 microseconds required for the addressing and transfer of a single data word. The program automatically resumes at the end of the transfer and continues till the next TPIX-I or TPOX-I trap request is received. Since the trap requests are normally received at close intervals, the computer is usually placed in a loop subroutine that repeatedly senses the state of the BIC controller.

The BIC, Meanwhile, is sensing the state of the peripheral controller. When the peripheral-controller buffer is again ready to accept or transmit data, the transfer sequence is again initiated and another word transferred to or from memory. These steps



T_0 is the start of the input sequence.

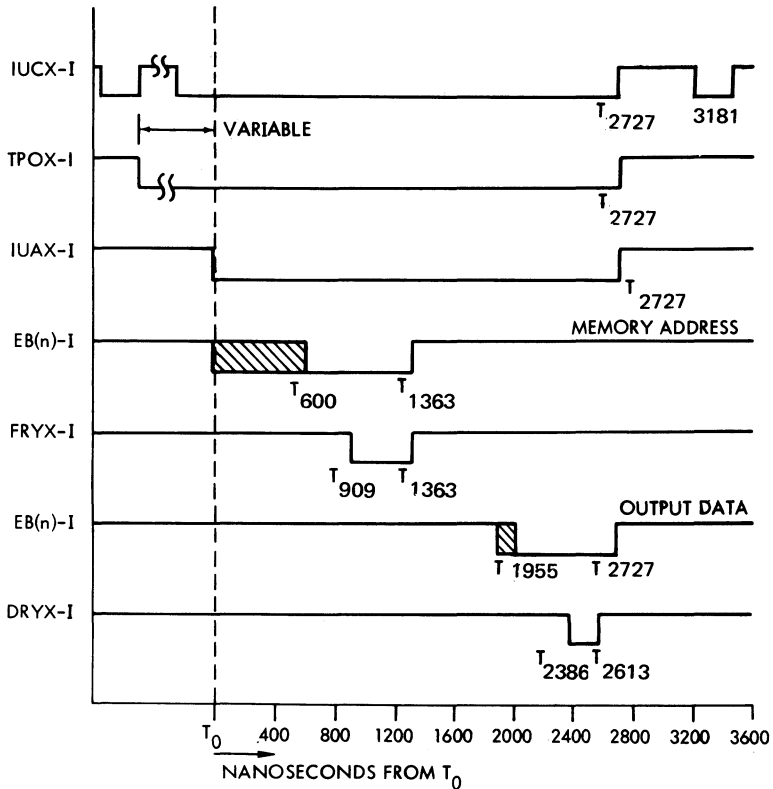
Logic levels: True = 0 vdc.
False = +3 vdc.

TPIX-I is normally off at T_{2727} ; it must be off by T_{2900} .

EB(n)-I (memory address) is normally on at T_0 ; it must be on by T_{600} . It is normally off at T_{1363} ; it must be off by T_{1500} .

EB(n)-I (input data) is normally on at T_{1350} ; it must be on by T_{1808} . It is normally off at T_{2727} ; it must be off by T_{2900} .

Figure 11-19. Trap-In Timing



T_0 is the start of the output sequence.

Logic levels: True = 0 vdc.
False = +3 vdc.

 = time when signal is settling.

TPOX-I is normally off at T_{2700} ; it must be off by T_{2900} .

EB(n)-I (memory address) is normally on at T_0 ; it must be on by T_{600} . It is normally off at T_{1363} ; it must be off by T_{1500} .

Figure 11-20. Trap-Out Timing

priority structure

are repeated until the initial-address register equals the contents of the final-address register. The BIC then generates an interrupt (normally through a PIM) to branch the CPU from its loop subroutine and back to the main program. A typical program for cycle-stealing I/O data transfer is given in Section 12.

Figures 11-19 and 11-20 detail the timing of the trap-in and trap-out cycle-stealing sequences.

Priority Structure

All three interrupt and trap request signals, IURX-I, TPIX-I, and TPOX-I, occur on the trailing edge of the interrupt clock, IUCX-I. It is possible, then, for two or more request-generating controllers or mainframe options to make simultaneous interrupt or trap requests.

To give priority to these requests, all of the request-generating controllers and mainframe options are linked on a priority chain (see Figure 11-1 for a typical example).

The priority chain consists of a series of interconnections, using the PR1X-I, PR2X-I, PR3X-I, and PR4X-I lines in the I/O bus. The high-priority end of the chain is grounded; the low-priority end is left open. The order of priority is established entirely by the physical order of these interconnections. (The priority of the peripheral

controllers connected to a PIM is established within the PIM. The priority of the PIM is established by its position in the priority chain.)

Figure 11-21 shows the priority logic within each controller on the chain. The input term (from the next higher priority controller) is PRMX-I. The output term is PRNX-I, which becomes the input term, PRMX-I, for the next controller on the chain.

When the request flip-flop within one controller is set (producing an interrupt or trap request on the trailing edge of the next IUCX-I interrupt clock), the PRNX-I output term is set to false, inhibiting the generation of interrupt or trap requests by any controller or mainframe option lower on the chain.

The PR1X-I through PR4X-I lines in the I/O bus are provided purely for convenience. They do not, in themselves, establish priority in any way. The lines are available on the backplanes of the mainframe and expansion chassis and in the I/O expansion cable between chassis. Figure 11-22 illustrates the use of the lines when the physical position of the controllers corresponds to their priorities. Figure 11-23 illustrates the use of the lines in a more complex situation. Figure 16-12, Section 16, shows the use of the I/O expansion cable to extend the priority chain from one chassis to another.

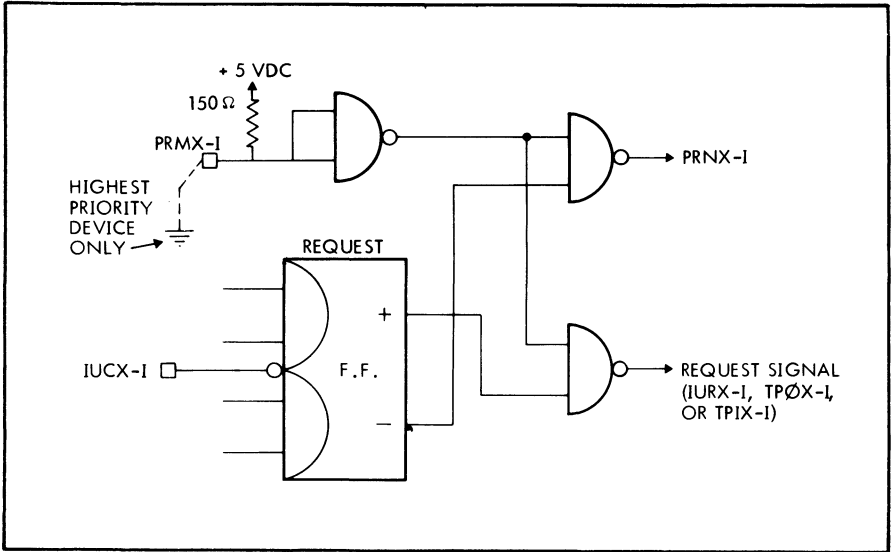
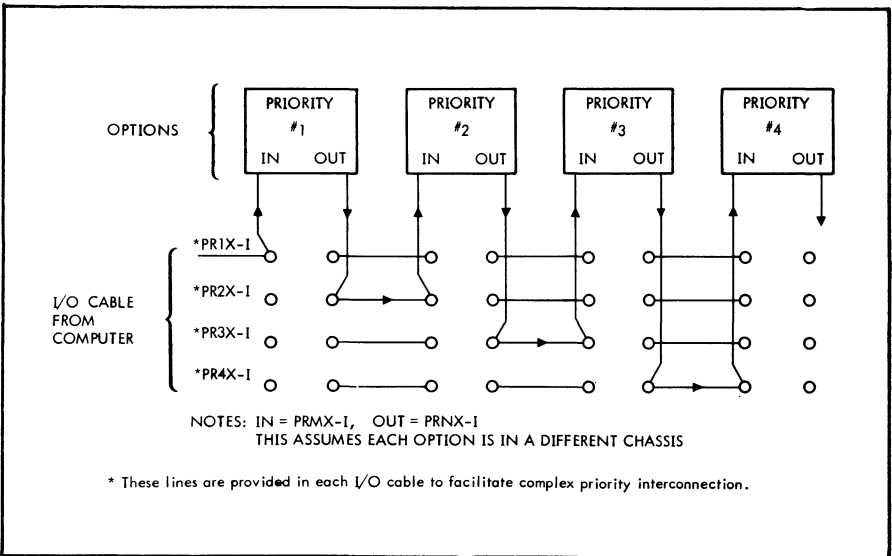
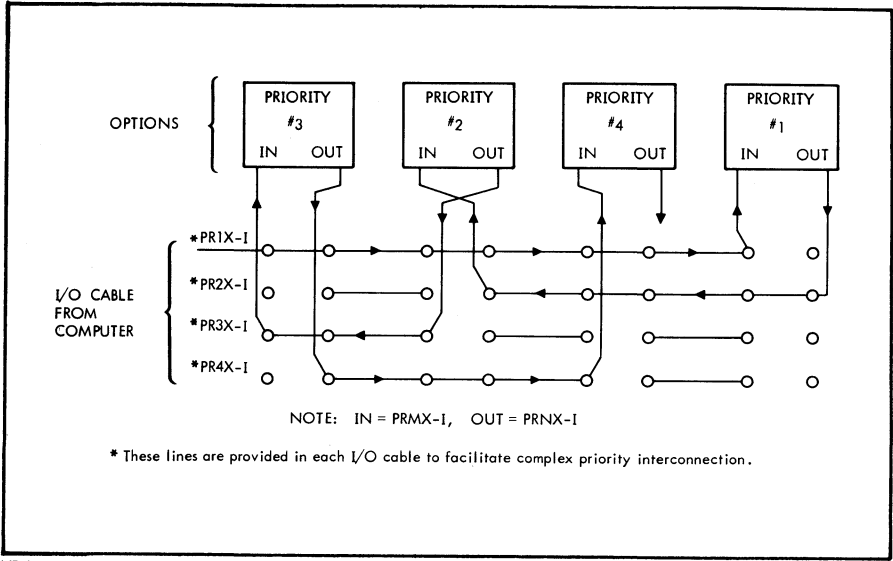


Figure 11-21. Priority Logic Within Controller



VT11-1244A

Figure 11-22. Device Interconnection: Device Position Corresponds to Priority Assignment



VTII-1245A

Figure 11-23. Device Interconnection: Device Position Does Not Correspond to Priority Assignment

SECTION 12 – INPUT/OUTPUT OPTIONS

There are three important options that can be added to the Varian 620/f I/O system: the Priority Interrupt Module to expand the priority interrupt capability, the Buffer Interface Controller to effect cycle-stealing, direct-memory-access (DMA) data transfers at rates up to 276,000 words per second, and the Priority Memory Access (PMA) option to effect data transfers at rates up to 1.33 million words per second.

Priority Interrupt Module (PIM)

The Model 620-16 priority-interrupt module (PIM) is an I/O option for the Varian 620/f computer that performs the following functions:

1. Establishes eight levels of interrupt priority for selected peripheral device controllers.
2. Stores interrupt requests originated by connected peripheral controllers, placing them on the I/O bus in order of their established priorities.

The PIM contains a line register, a sync register, a mask register, a priority logic and control section, and an interrupt address generator.

Installation of the PIM establishes a "priority within a priority" system. Peripheral controllers not normally capable of generating an interrupt address can do so when connected to the PIM. Priority levels for these controllers are established by their connections to the interrupt cable. In addition, the PIM provides entry to a

memory location predetermined for each of the controllers connected. The PIM also stores interrupt requests from its associated peripheral controllers, processing each in order of its priority via the computer I/O bus.

The PIM can mask interrupts from selected controllers when so programmed. In addition, input devices are prevented from monopolizing computer time to the exclusion of the main program.

The PIM is on one 7-3/4-by-12-inch etched-circuit card (DM124-1), which is installed in any I/O connector. Interface with the central processor is through the computer backplane wiring or through the I/O cable. Peripheral device controllers interface with the PIM through an interrupt cable that is connected to edge connectors on the PIM card or through a PIM connector. Controller priority levels are normally hardwired in at the time of installation.

PIM Functional Description

Communication between the PIM and the 620/f is similar to that of any peripheral controller except that the PIM can request a program interrupt. When the computer acknowledges the interrupt, the PIM specifies the memory location of the instruction to be executed by the 620/f.

The PIM automatically scans the interrupt

Characteristic	Specification
Size	One 7-3/4-by-12-inch etched-circuit card (DM124-1).
Organization	Contains three 8-bit registers (line, sync, and mask), an interrupt address generator, a priority logic and control section, and necessary input receivers and output drivers.
Interconnection	Interfaces with 620/f through the mainframe backplane or through the 620/f I/O cable . Connects to peripheral device controllers through the interrupt cable or via the backplane.
Connectors	One 122-terminal card-edge mating with female at backplane. Two 44-terminal card-edge mating with females on the interrupt cable.
Interrupt Control Capability	Eight priority levels, customer assignable.
I/O Capability	Five external control (EXC) and three transfer commands.
Type of Interrupt	False-to-true level transition.
Interrupt Line Scan	Once every 900 nsec.
Input Power	+5V dc.
Logic Levels:	
I/O Cable	Negative Logic: True 0.0 to +0.45 V dc. False +2.8 to +3.6V dc.
Interrupt Cable	Negative Logic: True 0.0 to +0.5V dc. False +2.5 to +5.0V dc.
Internal	Positive Logic: True +2.4 to +5.5V dc. False 0.0 to +0.5V dc.
External Device Address	PIM = 04X, where X = 0 to 7 hardwired, customer assignable.

Figure 12-1. Priority Interrupt Module Specifications (1 of 2)

Memory Locations Required	two per connected interrupt line, consecutive pairs, maximum 16.
Interrupt Address Location	Anywhere in memory locations 0100 through 0177
PIM Priority	Customer assigned. Determines placement in the priority chain.
Peripheral Controller Priority	Eight customer-assigned levels. Determines hardwired connection to the interrupt cable.
Operational Environment	0 to +50 degrees C, up to 90% relative humidity without condensation.

Figure 12-1. Priority Interrupt Module Specifications (2 of 2)

lines every 900 nanoseconds. If signals are detected on more than one line, the highest-priority signal is acknowledged. The remaining interrupt requests are stored in the PIM interrupt register until acknowledged. The PIM has an eight-bit mask register for disarming any or all of the eight interrupt lines. This register is set to zero's to arm interrupts. It is set to one's to disarm interrupts.

Acknowledgment of an interrupt by the central processor causes execution of the instruction located at the memory address specified by the PIM. The instruction can be any 620/f command except an I/O command. An interrupt is thus serviced in one instruction period.

If the executed instruction is jump and mark, the PIM interrupt system is automatically inhibited. This state is terminated only by the program.

The PIM interrupt capability is also inhibited:

- During any two-word instruction.
- During any I/O instruction.
- Until at least one non-I/O instruction has occurred after an I/O instruction.
- During any shift instruction.
- During a shift that occurs as part of a multiplication or division operation.
- Immediately following a shift operation during which a trap occurred.
- During a manual step.
- When in step mode (i.e., when halted).
- During manual or programmed halt.

Mnemonic	Program Code	Functional Description
A. External Control		
EXC 014X*	10014X*	Clear interrupt registers
EXC 024X	10024X	Enable PIM
EXC 034X	10034X	Clear interrupt registers and enable PIM
EXC 044X	10044X	Disable PIM
EXC 054X	10054X	Clear interrupt registers and disable PIM
B. Data Transfer		
OME 04X	10304X	Transfer memory to mask register
OAR 04X	10314X	Transfer A register content to mask register
OBR 04X	10324X	Transfer B register content of mask register
*X represents the last character of the external device address (0through 4 or 6 as determined by jumper connections on the backplane).		

Figure 12-2. PIM Reserved Address/Instruction Codes

- j. Until the run mode has been fully entered.
- k. During an execute command (conditional or unconditional), including the time required for execution of the instruction specified by the command.

When program loops contain only uninteruptable instructions, interrupts cannot occur. When recognition of an interrupt is imperative (such as power failure/restart interrupts), at least one NOP must be added to such loops. If placed immediately after two or more EXC instructions, two NOP's are required.

If the computer contains the Memory Protect option, the interrupt can be recognized after any instruction except EXC.

PIM Programming

The PIM has no operating controls. Its functions are directed by a program written by the user.

Figure 12-2 gives the command codes applicable to the PIM.

When preparing a PIM program, clear the interrupt registers to establish initial conditions. To mask peripheral controllers, write a mask word in the program. (The eight least significant bits of the mask word correspond to the eight priority interrupt lines. Setting bit 0 inhibits the highest

001000		STRT	,ORG	,01000	
001000	011011		,LDA	,MASK	FETCH INTERRUPT MASK
001001	103140		,OAR	,040	AND STORE IN REGISTER
001002	006010		,LDAI	,0377	INITIALIZE OUTPUT DATA
001003	000377				
001004	103137		,OAR	,037	PRIME PAPER TAPE PUNCH
001005	100240		,EXC	,0240	ENABLE PIM
001006	005000		,NOP	,	
001007	001000		,JMP	, *-1	DELAY FOR INTERRUPTS
001010	001006				
001011	000376	MASK	,DATA	,0376	
*		INTERRUPT PROCESSING SUBR			
001012	000000	INTR	,ENTR	,	
001013	005311		,DAR	,	DECR OUTPUT DATA
001014	103137		,OAR	,037	OUTPUT DATA TO PUNCH
001015	100240		,EXC	,0240	RE-ENABLE PIM
001016	001010		,JAZ	,*+4	
001017	001022				
001020	001000		,JMP*	,INTR	EXIT
001021	101012				
001022	100440		,EXC	,0440	DISABLE PIM
001023	000000		,HLT	,	END OF PROGRAM
		INTERRUPT ADDRESS			
000100			,ORG	,0100	
000100	002000		,JMPM	,INTR	
000101	001012				
	000000		,END	,	

Figure 12-3. Sample Program Using the PIM

priority line, setting bit 1 inhibits the second-highest priority line, setting bit 2 inhibits the third-highest priority line, etc.) The mask register must be loaded by the program after any power-up sequence, including the power-up cycle of the power failure/restart option, if any. System reset does not clear the PIM mask register.

In any instruction sequence of noninterruptable instructions that is longer than the shortest service interval of the attached peripherals, insert an NOP (two NOP's

following the two or more EXC instructions) to allow the PIM to request an interrupt if necessary. This is mandatory if the system contains a Model 620/L-14 power failure/restart.

Figure 12-3 gives a simple program in which 256 binary frames are transmitted to the high-speed paper tape punch using the PIM. The program covers memory locations 01000 to 01023, using DAS symbols with corresponding machine-language octal codes.

Mnemonic	Name	Description
EB00-I through EB 15-I	E Bus Signals	Used to transmit address and function codes between the CPU and the PIM
DRYX-I	Data Ready	Used in the PIM to reset the DTOX flip-flop.
FRYX-I	Function Ready	Indicates the E bus contains address information. The type of address depends upon the state of IUAX-I: When IUAX-I is true, a memory address is on the E bus from the address generator of the PIM. When IUAX-I is false, a device address with an associated function code is on the E bus from the CPU.
IUAX-I	Interrupt Acknowledge	Indicates the PIM interrupt is acknowledged. Causes the memory address generated by the PIM address generator to be placed on the E bus.
IUCX-I	Interrupt Clock	A 1.1 MHz clock signal that provides timing synchronization of the functions within the PIM. This clock is in a true state whenever IUAX-I is true.
IUJX-I	Interrupt Jump	Inhibits the PIM after a jump and mark instruction by resetting the PRME flip-flop.
IURX-I	Interrupt Request	Output level from the PIM indicating a request for an interrupt.
PRMX-I	Priority In	Input on the priority chain via circuits of all controllers having a higher priority than the PIM
PRNX-I	Priority Out	Output from the PIM priority circuit to all controllers having a lower priority. (NOTE: This signal does not apply to peripheral controllers connected to the PIM interrupt cable.)
SYRT-1	System Reset	An input from the CPU to reset the control logic in the PIM.

Figure 12-4. I/O Control Signals Between CPU and PIM

PIM Timing

Since the PIM serves as an interrupt-interface between a peripheral device and the CPU, the processing and timing of an interrupt request by the peripheral device is in two phases.

The first phase involves the priority logic within the PIM itself. If two or more requests are received simultaneously (during the same interrupt-clock pulse), the lower priority requests are stored until the higher priority requests are processed.

The second phase is the generation of an interrupt request by the PIM and the processing of the request by the CPU. The CPU views the PIM as simply another controller connected to the I/O bus, with a priority established by its priority-line interconnections. If higher priority interrupts are being processed, the PIM request is inhibited.

Once the PIM interrupt request to the CPU has been made and acknowledged, the processing and timing of the request is the same as that of the generalized case described in Section 11.

PIM Address Assignments

External device addresses 040 through 044 and 046 are reserved for the PIM's, with 040 normally assigned to the first PIM, 041 to the second, etc. The device address for each PIM is established by jumper wire connections on the backplane.

Interrupt Memory Locations

Each interrupt line is assigned two sequential memory locations. These can be anywhere within locations 0100 through 0177.

The addresses are established by wiring E104, E105, and E106 on the backplane

connector to ground, depending on the memory locations used. The even-numbered addresses must always be placed on the E bus by placing a zero in bit 0 of the memory address word.

The interrupt address generator consists of coding logic that generates the binary number of the interrupt line requesting the interrupt. The outputs of the three generator gates correspond to 000 through 111, placed on the E bus in positions 0 through 2.

PIM/CPU Interconnections

Connection to the central processor is hard-wired through the computer mainframe backplane. All signals enter or leave the PIM through receiver or driver stages that provide buffering between internal circuits and external lines. Figure 12-4 summarizes the I/O signals between the CPU and PIM.

PIM/Controller Interconnections

For peripheral controllers installed in the same chassis as the PIM, the interrupt lines are connected at the backplane. For controllers in other chassis, an interrupt cable is required. Card-edge connectors J1 and J2 on the PIM are parallel-wired, permitting connection of two interrupt cables. Figure 12-5 gives the pin assignments for the interrupt lines.

The PIM interrupt cable has eight lines for connection with up to eight peripheral controllers. These lines (labeled IL00-through IL07-, where IL00 has the highest priority) can be connected to the controllers in any order of priority.

The cable is either 10 or 20 feet long. The PIM is the termination for the interrupt

cable, with no external termination shoe required.

Interrupt Lines	P1	J1	J2
IL00-	108	3	3
IL01-	114	13	13
IL02-	104	9	9
IL03-	110	15	15
IL04-	102	11	11
IL05-	88	5	5
IL06-	112	1	1
IL07-	86	7	7

**Figure 12-5. PIM Pin Assignments
for Interrupt Lines**

Buffer Interface Controller (BIC)

The Model 620-20 Buffer Interface Controller (BIC) is an I/O option for the Varian 620/f computer.

The function of the BIC is to free the central processing unit to perform other program functions during block word transfers. In order to do this, the BIC transfers 16-bit words to and from memory and peripheral device controllers on a cycle-stealing, direct-memory-access (DMA) basis. The transfers can occur up to a maximum rate of 276,000 words per second; the typical transfer rate without the BIC is 30,000 words per second.

A single circuit card contains the entire BIC. The BIC plugs into the mainframe or an expansion chassis of the computer.

The BIC is used to control magnetic tape and disc devices, card and paper tape readers

and punches, and analog-to-digital controllers. Up to ten of these peripheral devices are operated by the BIC under program control. A computer system may include as many as four BICs.

The BIC is a system priority device; however, the peripheral devices connected to it have no priority of their own.

BIC Functional Description

The BIC is functionally divided into two circuits: address registers and sequence control.

Two address registers contain the memory locations of output data or input data, depending on the command input or output.

The initial (I) register stores the address of the first input or output word. This register is incremented after each data word transfer. When the block word transfer is complete, the I register contains the address of the last data word to be transferred.

The final (F) register stores the address of the last word to be transferred. Unless the peripheral device is abnormally stopped, the address in the F register will be the same as the address in the I register when word transfer is ended. When I and F registers reach comparison, the block word transfer is complete.

The sequence control circuit generates the control signals which coordinate address and data transfer between the CPU and the BIC and peripheral device controllers. The data is not routed through the BIC but are directly transferred between the device controller and memory.

Under program control, the CPU senses that the BIC is not busy and prepares the BIC to

Parameter	Description
Organization	Contains input receivers and output drivers, two 15-bit address registers and a sequence control circuit
Control capability	Up to ten device controllers
I/O capability	Two external control commands Eleven transfer commands Two sense commands
I/O transfer rate	Synchronized to peripheral device rate; maximum 276,000 words per second
I/O signal limits (rise/fall)	Minimum 10 nanoseconds; maximum 100 nanoseconds
Logic levels	
To the I/O bus	Negative logic True: 0.0 to +0.5 Vdc False: +2.8 to +3.6 Vdc
Internal	Positive logic True: +2.4 to +5.5 Vdc False: 0.0 to 0.5 Vdc
Size	One 7-3/4-inch-by-12-inch etched-circuit card
Interconnection	Interfaces with I/O cable through backplane connector
Connectors	One 122-terminal card-edge connector (mates with female connector at backplane)
Input power requirement	+5 Vdc

Figure 12-6. Buffer Interlace Controller (BIC) Specifications

BIC

receive the initial and final data addresses. The CPU then senses that the selected peripheral device is not busy and loads the I and F registers. At the same time, the BIC is activated and the peripheral controller is started. The BIC then assumes control of the data transmission, allowing computer operational registers to be used by the program for other functions.

Data transfer is accomplished between memory and the device controller via the E bus in the I/O cable. The BIC counts the words transferred and, when the data block transfer is complete, disconnects the device controller and assumes a not busy state. Data transfer may also be terminated upon request from the peripheral device controller.

The physical, electrical, and operational specifications of the BIC are listed in Figure 12-6.

BIC Programming

There are no operating controls or indicators on the BIC.

The BIC responds to the commands listed in Figure 12-8. Two device addresses are assigned to each BIC to differentiate functions directed by the I/O instruction. Addresses 020 through 027 are reserved for BICs. Address/instruction codes in Figure 12-8 are for the first BIC in a system. If additional BICs are installed, the addresses should be incremented by two for each additional BIC (i.e., second BIC addresses should be 022 and 023).

When preparing a program for use with the BIC, the programmer should sense the status of the BIC and the desired peripheral controller. After a not busy response is received from both controllers, they should be initialized. Then the BIC address registers

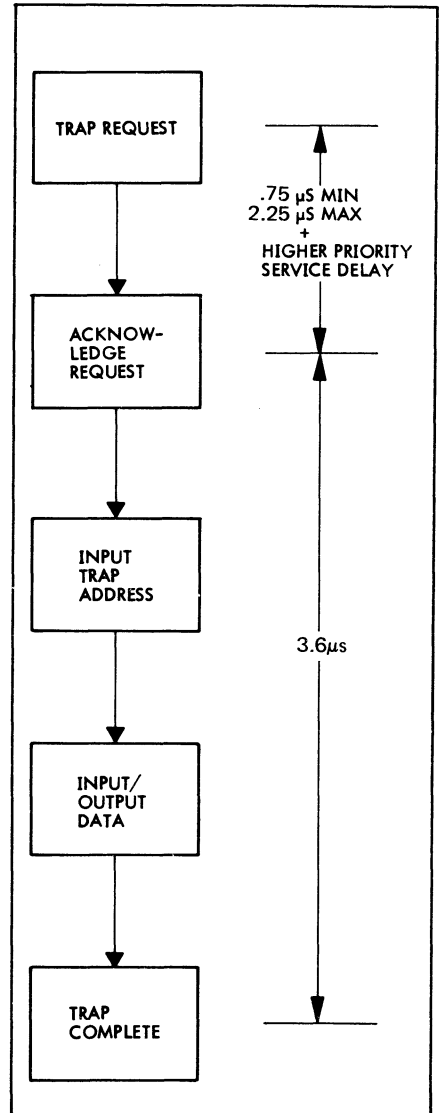


Figure 12-7. BIC-Controlled, Cycle-Stealing Trap Sequence, General Flow

Mnemonic	Octal	Functional Description
External Control		
EXC 020	100020	Active Enable
EXC 021	100021	Initialize
Transfer		
OAR 020	103120	Load Initial Register from A
OBR 020	103220	Load Initial Register from B
OME 020	103020	Load Initial Register from Memory
OAR 021	103121	Load Final Register from A
OBR 021	103221	Load Final Register from B
OME 021	103021	Load Final Register from Memory
INA 020	102120	Read Initial Register into A
INB 020	102220	Read Initial Register into B
IME 020	102020	Read Initial Register into Memory
CIA 020	102520	Read Initial Register into Cleared A
CIB 020	102620	Read Initial Register into Cleared B
Sense		
SEN 020	101020	Sense BIC Not Busy
SEN 021	101021	Sense Abnormal Device Stop

Figure 12-8. Buffer Interlace Controller (BIC) Commands

001000			,ORG	,01000	
001000	101020	BIC0	,SEN	,020,BIC1	CK BIC NOT BUSY
001001	001007R				
001002	100401		,EXC	,0401	INIT-TTY
001003	100021		,EXC	,021	INIT BIC
001004	005000		,NOP	,	
001005	001000		,JMP	,*-5	
001006	001000R				
001007	101101	BIC1	,SEN	,0101, BIC2	CK TTY WRITE READY
001010	001014R				
001011	005000		,NOP	,	
001012	001000		,JMP	,*-3	
001013	001007R				
001014	103120	BIC2	,OAR	,020	SET BIC I-REG
001015	103221		,OBR	,021	SET BIC F-REG
001016	100020		,EXC	,020	ACTIVATE BIC
001017	100101		,EXC	,0101	CONNECT WRITE REG
001020	101020		,SEN	,020,BIC3	CK BIC NOT BUSY
001021	001025R				
001022	005000		,NOP	,	
001023	001000		,JMP	,*-3	
001024	001020R				
001025	101021	BIC3	,SEN	,021,BIC5	CK ABN STOP
001026	001032R				
001027	007400		,ROF	,	
001030	102520	BIC4	,CIA	,020	INPUT BIC I-REG
001031	000000		,HLT	,	
001032	007401	BIC5	,SOF	,	SET ABN FLAG
001033	001000		,JMP	,BIC4	
001034	001030R				
	000000		,END	,	

Figure 12-9. Typical BIC Service Routine

NOTE: Routine should include instructions to prime TTY

Mnemonic	Name	BIC Function
ACEX	Activate enable flip-flop	Stores activation of BIC
ADSX	Abnormal device stop flip-flop	Stores end of data from peripheral controller
BCAX	Buffer controlled activate flip-flop	Stores the activation of the BIC and the peripheral device controller
BCDX	Buffer controller deactivate	Initiates termination of data transfer by the peripheral device controller
CDCX	Controlled device connected	Indicates that the peripheral device is connected
CIRX	Clear I register	Resets the flip-flops in the I register
CLEX	Clock enable	Enables incrementation if the I register
CLEZ	Clock Enable	Enables end of data sequence
DCEX	Device connect enable	Enables selection of a peripheral device
DESX	Device stop flip-flop	Stores the requirement to stop the peripheral device
DRYX	Data ready	Indicates the E bus contains a word of data
DSTX	Device stop permit flip-flop	Stores the end of the data transfer
EBD1	E bus drive 1	Enables loading of the F register
EBDX	E bus drive flip-flop	Stores the need to initially load the F register from the E bus
EBii	E bus bit	Data or address to be transferred
EBii	E bus bit inverted	Part of the BIC device address
FRYX	Function ready	Indicates the E bus contains an address

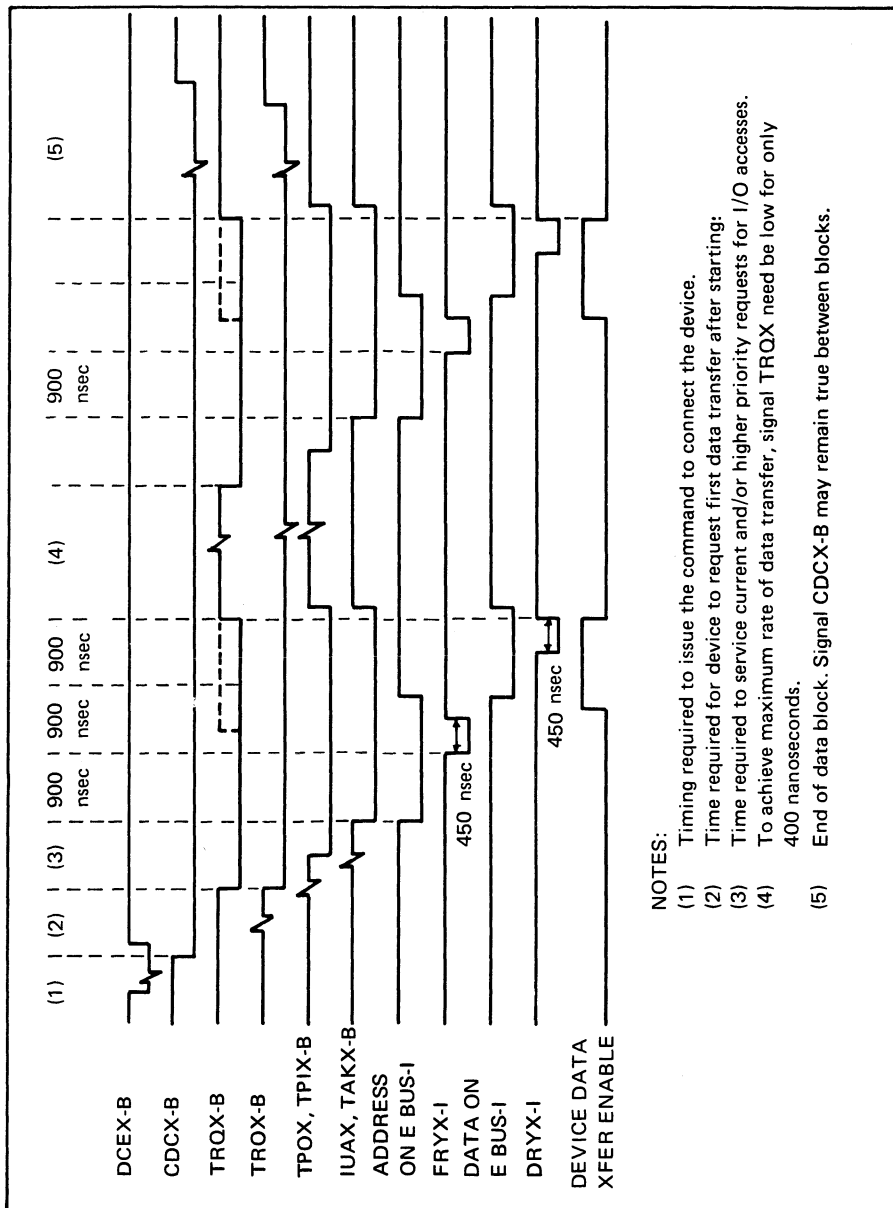
Figure 12-10. BIC Mnemonic Definitions (1 of 3)

Mnemonic	Name	BIC Function
FiiX	F register flip-flop	Stores final data address bit
IEFX	Initial equals final	Indicates content of I register is equal to content of F register
IEF2	Initial equals final 2	Initiates deactivation of the BIC and the peripheral device controller
IFMX	Initial equals final memory flip-flop	Stores the incrementation of the I register to the value of the F register
INIT	Initialize	Sets and resets BIC flip-flops to their initial condition
INTX	Interrupt	Requests interrupt (BIC stopped)
IUAX	Interrupt acknowledge	Enables servicing of BIC-connected peripheral device controllers
IUCX	Interrupt clock	Provides timing for servicing BIC
liiX	I register flip-flop	Stores initial and subsequent data address bits
LFRX	Load F register	Gates E bus address into F register
LFXX	Load F register permit flip-flop	Stores command to load F register
LIRX	Load I register	Gates E bus address into I register
LIXX	Load I register permit flip-flop	Stores command to load I register
PRMX	Priority input	Gives priority to BIC

Figure 12-10. BIC Mnemonic Definitions (2 of 3)

Mnemonic	Name	BIC Function
PRNX	Priority output	Passes priority to next in line after BIC is serviced
OIRX	Output I register	Gates contents of I register onto E bus
RIXX	Read I register flip-flop	Stores requirement of central processor to know contents of I register
SERX	Sense response	Indicates whether the BIC is busy
SYRT	System reset	Generates initialize signal when SYSTEM RESET switch is pressed
TAKX	Transfer acknowledge	Indicates that requirements for data transfer have been met
TAOX	Trap address out flip-flop	Stores the placing of the data address on the E bus
TCOX	Trap command flip-flop	Stores the need for a trap requested by the BIC
TPDX	Trap detect flip-flop	Stores the peripheral device request for a trap when data is to be transferred
TPIX	Trap input	Indicates that the BIC is ready to transfer data to memory
TPOX	Trap output	Indicates that the BIC is ready to transfer data from memory
TROX	Transfer out	Indicates the direction (in or out) of data transfer
TRQX	Transfer request	Indicates that the peripheral device is ready for the data transfer

Figure 12-10. BIC Mnemonic Definitions (3 of 3)



NOTES:

- (1) Timing required to issue the command to connect the device.
- (2) Time required for device to request first data transfer after starting:
- (3) Time required to service current and/or higher priority requests for I/O accesses.
- (4) To achieve maximum rate of data transfer, signal TRQX need be low for only 400 nanoseconds.
- (5) End of data block. Signal CDCX-B may remain true between blocks.

VT11-1246

Figure 12-11. BIC Trap Sequence Timing

should be loaded with the initial and final memory addresses of the block of data to be transferred, a BIC activate enable instruction placed on the I/O cable, and the transfer started. Although the program requires loops for use with sense instructions and to handle abnormal conditions, transfer of the data block is accomplished by the BIC without further program instructions.

Figure 12-9 shows a typical service routine for the BIC, a teletype paper tape punch operation under BIC control. Using DAS symbols with corresponding machine language octal codes, the program covers memory locations 01000 through 01034.

Once the program is loaded, the operator must insert the initial punch buffer address into the A register and the final address into the B register for each run. The program when started will: initialize the BIC and teletype punch, initiate the transfer, read the contents of the BIC initial register into the A register at completion of the transfer, set the overflow indicator if the termination was abnormal, and then halt. The punch buffer must contain only ASCII characters; the first character must be 0222 (punch on) and the last character must be 0224 (punch off).

The mnemonics used in the BIC are listed alphabetically in Figure 12-10. A brief description of each signal's function is given as well as the proper signal name.

BIC Timing

The sequence and timing of a BIC controlled cycle-stealing data transfer are shown in Figures 12-7 and 12-11.

BIC Interconnections

The BIC is on an etched-circuit card, 7-3/4 by 12 inches, designated DM126.

All connections to the BIC are made through the 122-terminal card-edge connector which mates with the corresponding backplane connector in the computer chassis or in an expansion chassis.

The BIC circuit card is normally located in a card slot in the I/O section of the same chassis as the peripheral controllers to which it is connected.

The BIC interfaces with the computer and peripheral controllers entirely via the backplane connectors. When two or more BICs are installed in the same chassis, the BIC control lines are only connected to the controller (or controllers) that each BIC is communicating with. No BIC control lines are installed between two BICs. When the controller is in a different chassis, control lines are extended via the B cable.

The BIC signal interfaces are listed in Figure 12-12. A circuit-card connector pin number follows each signal mnemonic. The definition of each mnemonic is given in Figure 12-10.

Input Signals			
Mnemonic	Pin	Mnemonic	Pin
BCDX-B	52	EB11-I	17
CDCX-B	54	EB12	18
DRYX-I	29	EB13	19
EB00-I	2	EB14-I	20
EB01	4,65	EB1I	68,69
EB02	6,70	EB2I	71,72
EB03	8	FRYX-I	27
EB04	10	INTX	75
EB05	11	IUAX-I	44
EB06	12	IUCX	45
EB07	13	PRMX	37
EB08	14	SYRT	43
EB09	15	TROX-B	50
EB10-I	16	TROX-B	49

Output Signals			
Mnemonic	Pin	Mnemonic	Pin
DCEX-B	56	EB09-I	15
DESX-B	60	EB10	16
EB00-I	2	EB11	17
EB01	4	EB12	18
EB02	6	EB13	19
EB03	8	EB14-I	20
EB04	10	PRNX	42
EB05	11	SERX-I	31
EB06	12	TAKX-I	58
EB07	13	TPIX-I	33
EB08	14	TPOX-I	35

Figure 12-12. BIC Input and Output Pin Assignments

Priority Memory Access (PMA)

The Model 620/f-12 Priority Memory Access (PMA) is a mainframe option for Varian 620/f computer systems. The PMA option interfaces with PMA controllers to provide logic functions to interface four data transfer channels with the Varian 620/f memory in a hardware-fixed priority. All signals are interlocked to free the PMA option/PMA controller interface from circuit and cable speed dependence, and to provide the capability to handle a variety of circuits and data rates. Specifications for the PMA option are given in Figure 12-13.

The PMA option can interface with up to eight PMA controllers distributed in any manner among the four channels although only one PMA controller per channel can be active at a time. Each PMA controller interfaces with the PMA option board that plugs into the Varian 620/f mainframe CPU tray and with the Varian 620/f I/O bus for program control (Figure 12-14).

PMA controllers are either designed by the user or supplied by Varian. A Varian-supplied block-transfer controller (PMA BTC) is described at the end of this section.

The four PMA option channels share the 15-bit address bus, the 16-bit bidirectional bus, the READ line, the SYRT-P line, the PRMS line, and the GO line. Each channel has its own request line (REQx) and its own acknowledge line (ACKx). Access to the PMA option control and bus signals is through J7 on the Varian 620/f mainframe backplane.

Control Card II in slot 10 of the mainframe CPU tray contains the memory access control logic which decides whether the PMA, DMA, or CPU gains access to memory. PMA has the highest priority, as shown in Figure 12-15.

PMA Functional Description

The PMA is functionally divided into six circuits: control, priority, data transfer, data drivers, input data receivers, and address receivers (Figure 12-16).

The control logic provides enable terms to the other functional blocks, priority check and full interlock control of the four channels, and control signal interface with the Varian 620/f memory access logic.

The priority logic assigns priority to the four PMA channels on a hardware basis. It controls the channel acknowledge lines and the PMA request (PMAR-) line to the Varian 620/f memory access logic.

The four channel request (REQx-) signals are gated directly to the memory access logic before priority is established to reserve the next memory cycle for the PMA. The priority logic constantly monitors the four channels until a request is received. The logic then establishes priority (channel 1 has the highest priority and channel 4, the lowest) and enables the proper acknowledge line. Upon receipt of the acknowledge signal, the PMA controller can disable the request or, if consecutive memory cycles are required, maintain the enabled condition.

The priority logic continues to monitor priority, thus, channel priority is reestablished between each PMA memory cycle upon receipt of Memory Data Read (MDR). If a higher priority channel remains enabled, all lower priority channels are locked out.

The data transfer logic controls the flow of data through the PMA option and synchronizes priority determination with memory cycle completion.

Sixteen data drivers (PD00+ through PD15+) provide gating and interface

Parameter	Description
Organization	Contains control logic, priority logic, data transfer logic, data drivers, and input data receivers and address receivers
Priority assignment	Exceeded only by the power failure/restart option
Number of channels	Four with hardware-assigned priority
Maximum number of controllers per PMA	Eight distributed in any manner among four channels (only one active controller per channel at a time)
Maximum cable length	20 feet
Typical maximum latency	1,085 nanoseconds
Typical minimum latency	378 nanoseconds
Typical maximum transfer rate	1.33 Mhz
Control signals	All signals are controlled by interlocked responses allowing true asynchronous operation
Logic levels	PMA bus Positive logic: True = +2.2 to +3.2V dc False @ -0.5 to +1.4V dc Negative logic: True = -0.5 to +1.4V dc False = +2.2 to +3.2V dc Internal Positive logic: True = +2.2 to +3.2V dc False = -0.5 to +1.4V dc
Size	One 3-by-15-inch (7.7 x 38.1 cm) wired-socket card
Interconnection	Plugs into the 620/f CPU motherboard and mounts the PMA bus connector on the 620/f backplane at location J7
Power consumption	+5V dc at 2.0 amperes -5V dc at 300 mA +3.2V dc at 2.2 amperes

Figure 12-13. PMA Specifications

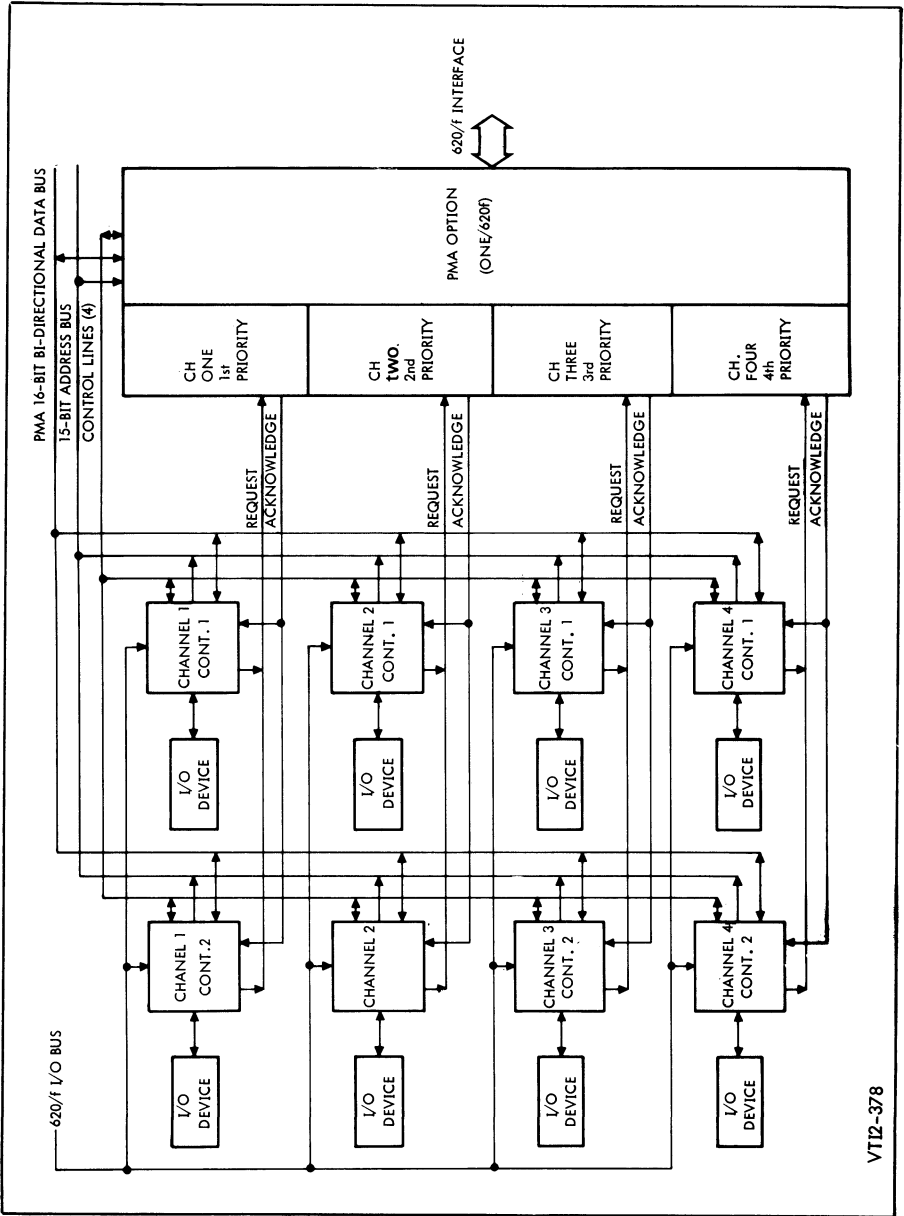
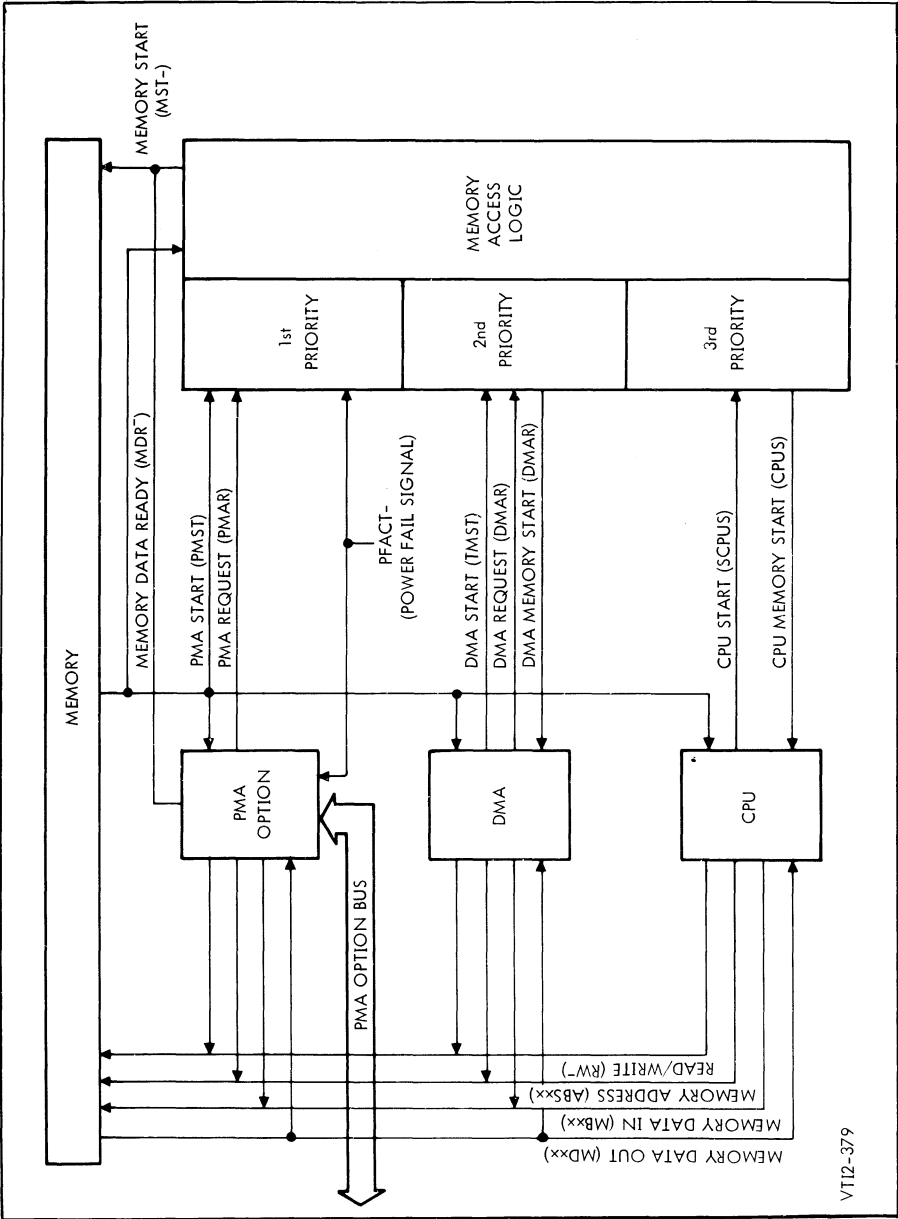


Figure 12-14. PMA Option/PMA Controller Interface



VT12-379

Figure 12-15. PMA, DMA, Control Line Out

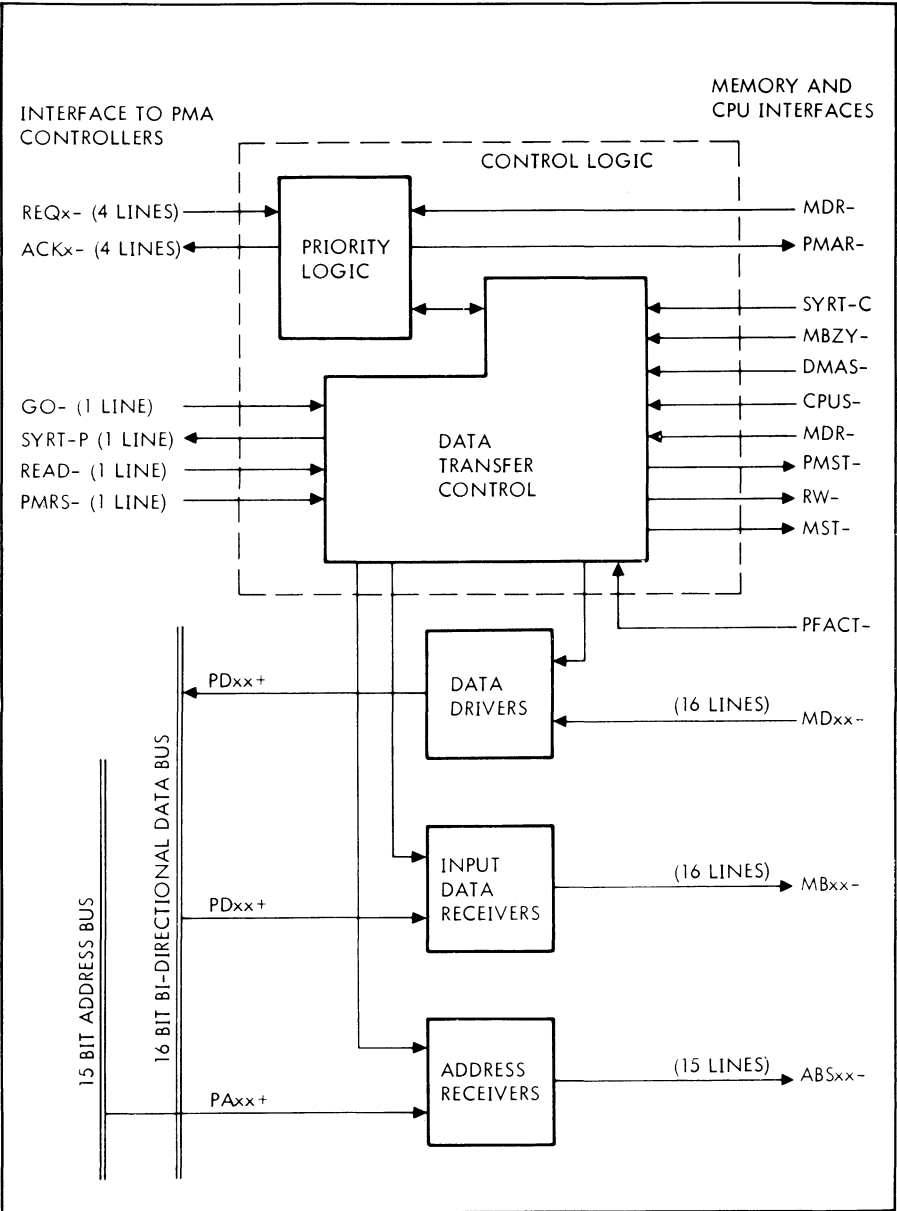


Figure 12-16. PMA Functional Block Diagram

Pin	Signal Designation	Description
1B	PA14+	Most significant bit

Pin	Signal Designation	Description
1B	PA14+	15-bit address bus
2B	PA13+	
3B	PA12+	
4B	PA11+	
5B	PA10+	
6B	PA09+	
7B	PA08+	
8B	PA07+	
9B	PA06+	
10B	PA05+	
11B	PA04+	
12B	PA03+	
13B	PA02+	
14B	PA01+	
15B	PA00+	
16B	SPARE	Least significant bit

17B	PD15+	16-bit bidirectional bus
18B	PD14+	
19B	PD13+	
20B	PD12+	
21B	PD11+	
22B	PD10+	
23B	PD09+	
24B	PD08+	
25B	PD07+	
26B	PD06+	
27B	PD05+	
28B	PD04+	
29B	PD03+	
30B	PD02+	
31B	PD01+	
32B	PD00+	
33B	SPARE	Least significant bit

Figure 12-17. PMA Connector Pin Assignments (1 of 2)

34B	READ-	
35B	GO-	
36B	SPARE	
37B	ACK1-	Acknowledgement signals for the four PMA channels
38B	ACK2-	
39B	ACK3-	
40B	ACK4-	
41B	PMRS-	PMA reset
42B	REQ1	Request signals for the four PMA channels
43B	REQ2-	
44B	REQ3-	
45B	REQ4-	
46B	SPARE	
47B	SYRT-P	System reset
48B	+3.2V	Termination voltage
49B	+3.2V	
50B	+3.2V	
1A	GND	
•	•	
•	•	
•	•	
50A	GND	

Figure 12-17. PMA Connector Pin Assignments (2 of 2)

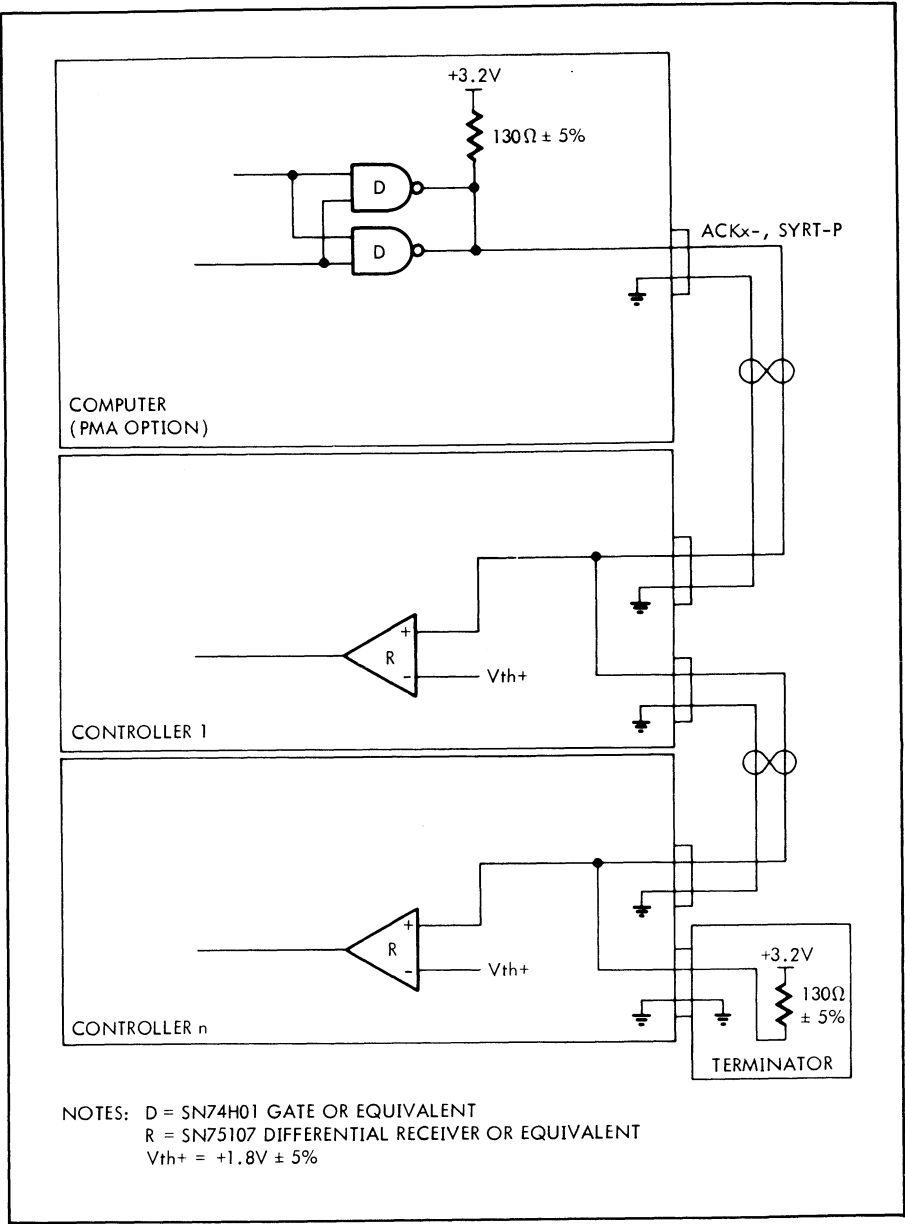
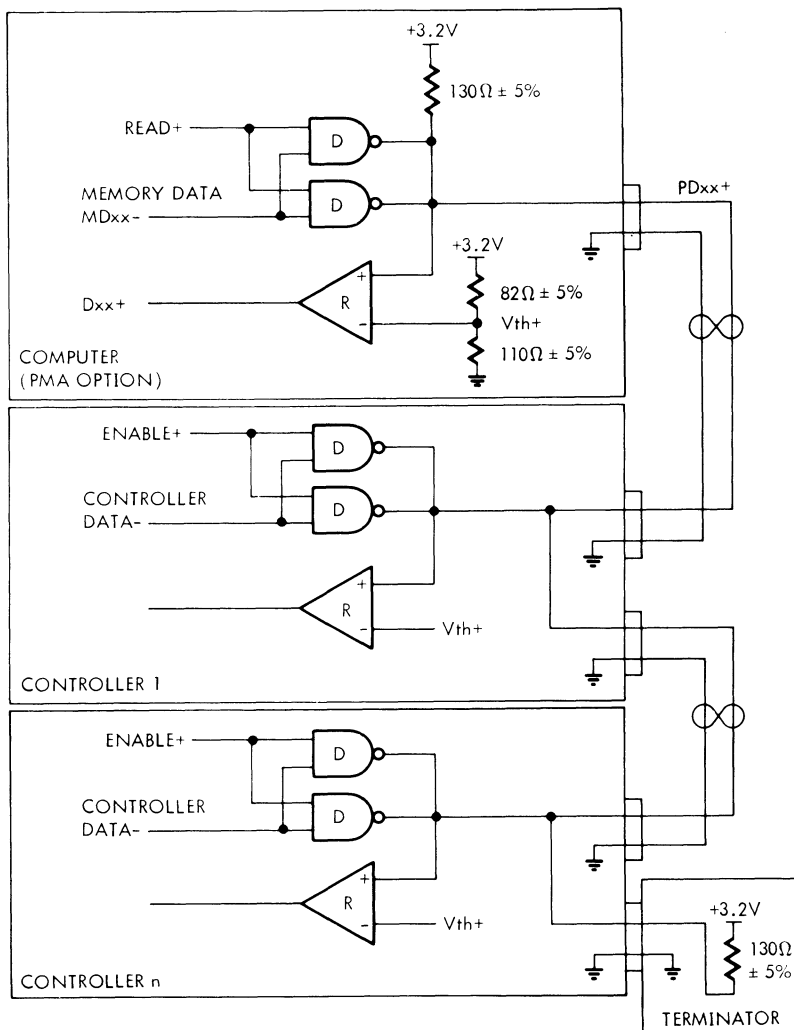


Figure 12-18. Typical PMA Control Line Out



NOTES: D = SN74H01 GATE OR EQUIVALENT
 R = SN75107 DIFFERENTIAL RECEIVER OR EQUIVALENT
 $V_{th+} = +1.8V \pm 5\%$

Figure 12-19. Typical PMA 16-Bit Bidirectional Bus Line

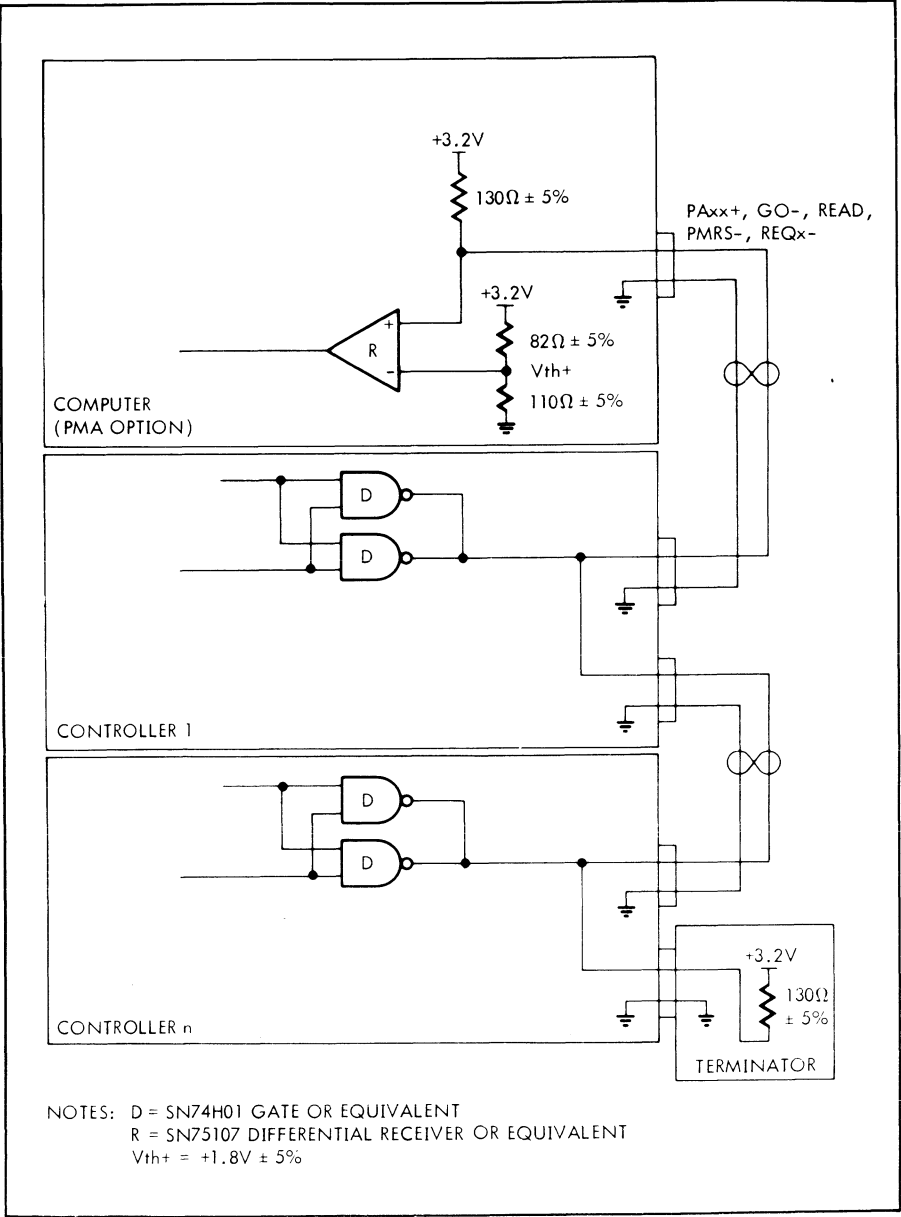


Figure 12-20. Typical PMA Control Line In or Address Line

between the memory output data bus and the PMA 16-bit bidirectional data bus.

Sixteen input data receivers (MB00- through MB15-) provide gating and interface between the memory input data bus (MBxx) and the PMA 16-bit bidirectional data bus (PDxx).

Fifteen address receivers (ABS00- through ABS14-) provide gating and interface between the memory address bus (ABSxx-) and the PMA 15-bit address bus (PAXx).

The physical, electrical, and operating specifications of the PMA option are listed in Figure 12-13.

PMA Option/Controller Interface

PMA controllers can be installed in any of the six I/O card slots of I/O expansion chassis I (Model 620/f-010).

PMA interconnection consists of routing signals from the PMA circuit board in card slot 2 of the CPU mainframe to the I/O groundplane in the expansion chassis. The expansion chassis groundplane is wired specifically for PMA use.

The PMA cable connects 100-pin card-edge connector J7 on the mainframe to circuit board connectors on the I/O groundplane of the expansion chassis. The PMA lines on the I/O groundplane must be terminated with a PMA termination shoe. The PMA cable is 15 feet long if the expansion chassis is located directly under the mainframe. The cable can be up to 20 feet long, if required. Approximately 2.4 feet of cable is used inside the expansion chassis.

In assignments for the 100-pin card-edge connector are given in Figure 12-17. The four PMA channels share the 15-bit address bus, the 16-bit bidirectional bus, the READ-

line, the SYRT-P line, the PRMS- line, and the GO- line. Each channel has its own request (REQx-) line and its own acknowledge (ACKx-) line.

PMA Option/Controller Interface Circuits

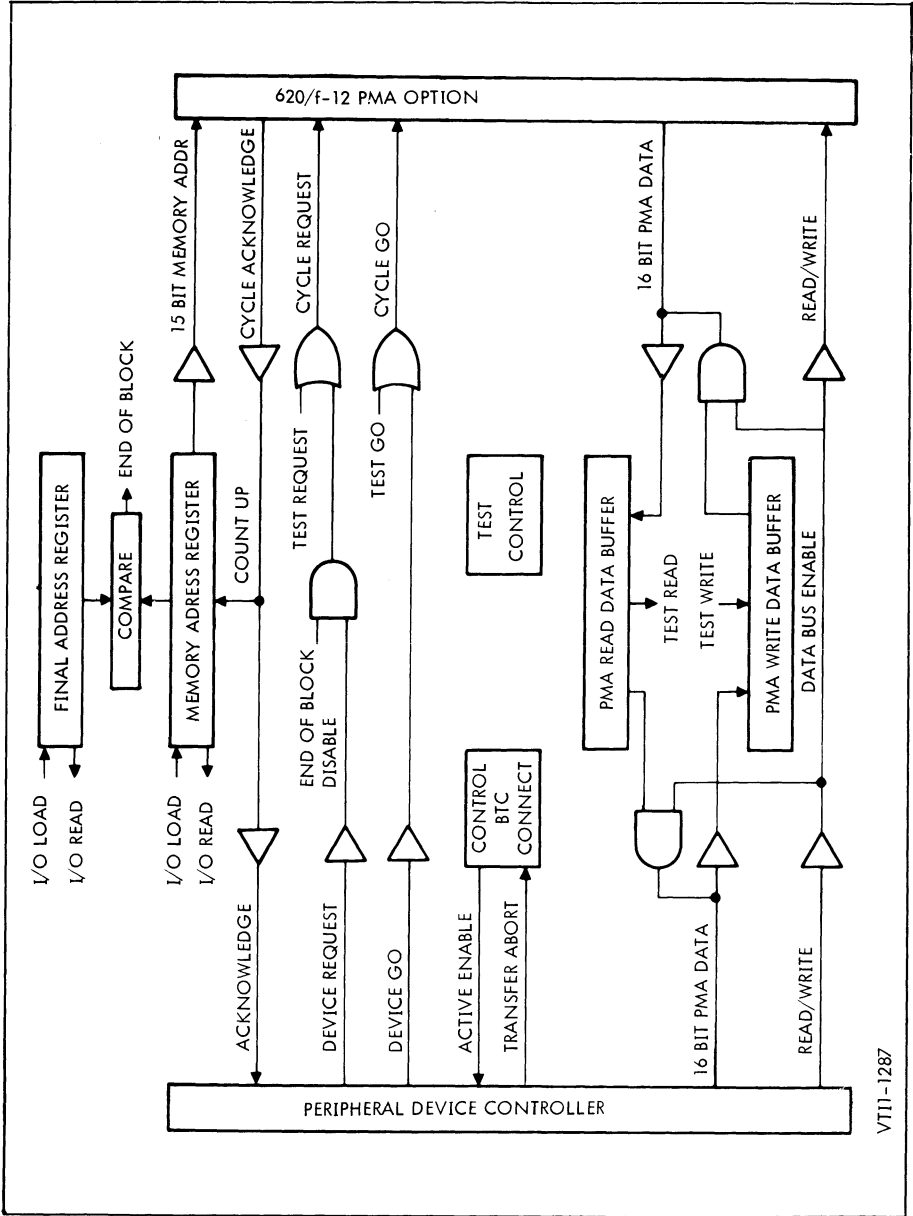
The PMA option/controller driver comprises two high-speed open-collector two-input NAND gates in parallel (SN74H01N). The parallel drivers are contained in the same integrated circuit package (part number 49A0042-000). User-designed controllers should also follow this practice to ensure equal current sharing by the paralleled drivers when sinking current to the ground voltage level.

The PMA option/controller receiver is a dual differential line receiver (SN75107N or equivalent; part number 49A0114-000). Refer to Figures 12-18, 12-19, and 12-20 for typical circuit descriptions.

The PMA cable is constructed with twisted-pair 24- or 26-gage wire and has a maximum length of 20 feet. A maximum of two controller chassis are packaged at a minimum distance apart to minimize discontinuity effects in the PMA cable. All lines in the cable are terminated at both ends. Termination resistors are mounted on the PMA option board, and termination for the other end of the cable is provided on the last controller or by a termination shoe located at the end of the cable.

PMA Block Transfer Controller (BTC)

The Priority Memory Access Block Transfer Controller (PMA BTC) interfaces the 620/f PMA option to a peripheral controller requiring fast access, high speed transfer of data in or out of processor core memory. The BTC provides memory address control when the transferred data is organized into "blocks" of 16-bit words.



VT11-1287

Figure 12-21. BTC Block Diagram

The BTC buffers one word of data in and out of the PMA option so that the peripheral controller timing requirements can be relaxed with respect to the fast PMA bus.

Peripheral controllers used with the BTC must be designed for PMA use since the PMA bus is separate from the E-bus shared by I/O and DMA.

As can be seen in Figure 12-21, the BTC interfaces to three points; the PMA option, the I/O bus, and the PMA controller.

Under program control, the BTC is loaded with the initial memory address and the final memory address of the block transfer. The BTC is then "connected" to the peripheral controller to enable it to connect to the PMA Port through the BTC.

A PMA transfer cycle is initiated by the peripheral controller issuing a PMA Request which the BTC passes on to the PMA option board in the CPU. When the PMA Option has captured the memory from the DMA and CPU, PMA Acknowledge is returned to the BTC and peripheral controllers. The memory then sits idle waiting for PMA GO from the peripheral controller.

When PMA GO is received at the PMA Option, the memory address and read/write line from the BTC are gated into the memory to start the cycle. Typically, the peripheral controller is ready to transfer data when it raises a PMA Request. This means it can "wrap-around" PMA Acknowledge as PMA GO with a minimum of delay.

As the PMA cycle occurs, the BTC buffers the transferred data, and increments its memory address counter for the next cycle. If the peripheral controller sustains its Request, it can "hog" the memory thru the PMA Port and receive consecutive memory cycles.

When the peripheral controller has received or sent enough words for its purpose, it removes its request to stop the PMA process. The BTC stops incrementing its memory address and CPU program resumes at the point where the peripheral request was acknowledged. The peripheral controller remains connected to the BTC.

When the accumulation of PMA word transfers or one long PMA transfer reaches the block length initially loaded into the BTC, the BTC unconditionally disconnects from the peripheral controller and PMA Port and generates a BTC finished term which can be used to set a PIM interrupt.

The BTC cannot be reconnected until it is reloaded with the parameters of the next transfer.

The peripheral controller may abort a transfer by initiating a disconnect to the BTC. The peripheral controller must inform the program of the reason for the abort so that the transfer may be redone in an orderly fashion.

Note: Any abort action by peripheral controller must not occur during a PMA/BTC memory cycle.

BTC Programming

PMA BTC Programming is identical to that of the DMA BIC except that the data transfer occurs much faster. Additional instructions are also provided, mainly to serve as diagnostic aids in testing the BTC, PMA bus and PMA option.

The PMA BTC instructions are given in Figure 12-22.

Like the DMA BIC, the PMA BTC uses a pair of device addresses — $2X$ and $2(X + 1)$, from the group 20 through 27. The possible even/odd address pairs are 20/21, 22/23, 24/25, and 26/27.

Execute Commands

EXC002X	100020	Activate Enable
EXC002 (X + 1)	100021	Initialize
EXC012X	100120	PMA Port Reset
EXC012 (X + 1)	100121	Test Data Transfer
EXC022X	100220	Test Cycle Write
EXC022 (X + 1)	100221	Test Cycle Read

Sense Commands

SEN002X	101020	Sense BTC Not Busy
SEN002 (X + 1)	101021	Sense Abnormal Device Stop
SEN012X	101120	Test Sense End of Block

Data Transfer Out

ØAR 2X	103120	Load BTC Initial Address Register ¹
ØBR 2X	103220	Load BTC Initial Address Register ¹
ØME 2X	103020	Load BTC Initial Address Register ¹
ØAR 2 (X + 1)	103121	Load BTC Final Address Register
ØBR 2 (X + 1)	103221	Load BTC Final Address Register
ØME 2 (X + 1)	103021	Load BTC Final Address Register

Data Transfer In

INA 2X	102120	Read Initial Address Register ²
INB 2X	102220	Read Initial Address Register ²
IME 2X	102020	Read Initial Address Register ²
CIA 2X	102520	Read Initial Address Register ²
CIB 2X	102620	Read Initial Address Register ²
INA 2 (X + 1)	102121	Read Final Address Register
INB 2 (X + 1)	102221	Read Final Address Register
IME 2 (X + 1)	102021	Read Final Address Register
CIA 2 (X + 1)	102521	Read Final Address Register
CIB 2 (X + 1)	102621	Read Final Address Register

¹ When immediately preceded by an EXC012 (X + 1) command (Test Data Transfer), the output commands will load the PMA Write Data Buffer from the E-Bus. The actual transfer clears the Test EXC logic. The buffer can then be transferred into memory by PMA with a Test Cycle Write command.

Figure 12-22. PMA BTC Commands (1 of 2)

² When immediately preceded by an EXC 012 (X + 1) command (Test Data Transfer), the input commands will read the contents of the PMA REad Data Buffer back over the E-Bus. The actual transfer clears the Test EXC logic. The buffer can be read in this fashion after a Test Cycle Read command has loaded the buffer through PMA.

Figure 12-22. PMA BTC Commands (2 of 2)

SECTION 13 — PERIPHERALS AND I/O INTERFACES

This section describes a sampling of various peripheral and I/O interface options that are available with the Varian 620/f computer.

The circuits of the Varian 620/f peripheral controllers are contained on 7.75-by-12-inch (19.7 x 30.3 cm) circuit cards that can be installed in any peripheral controller slots in the Varian 620/f mainframe or expansion chassis.

The Model Number listing in Section 6 indicates whether the controller occupies the space of one card slot (PC slot) or three card slots (U slot).

Interconnection between the controller and the peripheral device is via a connector on the rear edge of the card.

The following list indicates the peripherals and I/O interfaces described in this section.

Device	Page
Teletypes	13-1
Paper Tape System	13-4
7-Track Magnetic Tape	13-7
9-Track Magnetic Tape	13-10
Disc Memories	13-13
Drum Memories	13-19
Line Printer	13-21
Oscilloscope Display	13-24
Punched Card	13-27
Digital Plotter	13-31
Buffer I/O	13-34
Relay I/O	13-37
Digital I/O	13-40
Analog Input	13-43
Analog Output	13-47

Analog Power Supply	13-50
Dataset Controllers	13-52
Automatic Calling Unit	13-59
Data Communication (DCC-1)	13-61
Varian 620/DC Communication System	13-63

Teletypes

The controller for the first teletype in a Varian 620/f system is mounted on a 3 x 15 inch card in an assigned slot in the mainframe chassis and connection between the computer and the TTY is via a connector on the back of the mainframe chassis (see Sections 15 and 16).

Additional TTY's require the addition of an optional TTY controller (Model 620/L-6, -7, -8) connected to the computer I/O bus.

The additional TTY's also require a TTY buffer board. Up to 8 TTY's can be included in a single computer system.

The factory-modified Teletype unit that is controlled by the TTY controller can be a model 33 ASR, 35 ASR, or 35 KSR. The ASR models include paper tape reader and punch facilities; the KSR model uses only keyboard-entered instructions and data.

Specifications of the optional TTY controller are given in Figure 13-1.

Organization	Contains input and output registers, timing control circuitry for simultaneous two-way transmission, and CPU/TTY interface logic
Peripheral Device	Factory-modified Teletype Model 33 ASR, 35 ASR, or 35 KSR, including cable
Speed	Operation controlled by TTY speed: 10 characters per second (100 milliseconds per character) at either random or sustained rate
Operation Modes	Input: From keyboard or paper tape reader Output: To TTY printer or paper tape punch
Device Address	TC 001
Sense Responses	Ready to read; ready to write
Memory Access Control	By CPU indirectly (requires TTY buffer board 01A0688-000) or by BIC
Types of Interrupt	Write ready and read ready interrupts available to PIM
Input Power	+5 V
Logic Levels	Postive Logic True: +2.4 to +5.5V dc False: 0.0 to +0.5V dc
Size	One 3.125 by 14.930 etched-circuit card
Interconnection	Interfaces with CPU, I/O bus, or BIC and TTY via mainframe backplane connector
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-1. Teletype Controller Specifications

Mnemonic	Octal	Function	
A. External Control			
EXC 101	100101	Connect Write Register to BIC	
EXC 201	100201	Connect Read Register to BIC	
EXC 401	100401	Initialize	
B. Transfer			
OAR 01	103101	Transfer A Register to Write Register	
OBR 01	103201	Transfer B Register to Write Register	
OME 01	103001	Transfer Memory Register to Write Register	
INA 01	102101	Transfer Read Register to A Register	
INB 01	102201	Transfer Read Register to B Register	
'ME 01	102001	Transfer Read Register to Memory Register	
CIA 01	102501	Transfer Read Register to Cleared A Register	
CIB 01	102601	Transfer Read Register to Cleared B Register	
C. Sense			
SEN 0101	101101	Write Register Ready	
SEN 0102	101201	Read Register Ready	
D. Teletype Command Codes			
Function	Symbol	Code	Typed As
Print Enable	SOM	201	Control and A
Print Suppress	EOT	204	Control and D
Reader On	XON	221	Control and Q
Punch On	TAPE	222	Control and R
Reader Off	XOFF	223	Control and S
Punch Off	TAPE OFF	224	Control and T

Figure 13-2. Teletype Instruction Codes

High-Speed Paper Tape Equipment

The Model 620-55 and 620-55A high-speed paper tape systems are composed of a controller card, a paper tape perforator, and a paper tape reader. A paper tape spooler is also available for use with the reader.

The controller card contains a data register which buffers the data words being transferred, a decoder section which interprets instructions received from the computer, a timing and control section which synchronizes operation of the peripheral equipment with the computer and necessary interface hardware.

The paper tape controller can transfer data from the tape reader to the computer; it can also transfer data to the tape perforator from the computer, or it can be used to reproduce paper tapes. The controller can transfer data into the computer in a continuous read mode, which places the tape reader in continuous slew until a stop command is received, or it can operate in a step read mode, requiring a new instruction from the computer for each transmitted data word.

Computer control of the paper tape system is accomplished through the I/O bus. The controller can also be operated under direction of the BIC.

Each controller is capable of operating one perforator and one reader.

Organization	Consists of a timing and control section, an instruction decoder, and an eight-bit data buffer register	
Control Capability	One tape reader and one tape perforator, operated on a time-shared basis	
I/O Capability	Five external control, eight transfer, and one sense instructions	
Logic Levels	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to 0.5V dc	
Operational Modes:	Model 620-55A	Model 620-55
Continuous Read	300 characters/sec	150 characters/sec
Step Read	1-300 characters/sec	1-150 characters/sec
Punch	1-75 characters/sec	1-75 characters/sec
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) etched-circuit card	
Interconnection	Interfaces with the computer and BIC via a 122-terminal connector; interfaces with tape punch and reader via two 44-terminal connectors	
Input Power	+5V dc at 540 mA	
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation	

Figure 13-3. High-Speed Paper Tape Controller Specifications

high-speed paper tape

Mnemonic	Octal	Function
A. External Control		
EXC 037	100037	Connect Punch to BIC
EXC 0437	100437	Stop Reader
EXC 0537	100537	Start Reader
EXC 0637	100637	Punch Buffer
EXC 0737	100737	Read One Character
B. Transfer		
OAR 037	103137	Load Buffer from A Register
OBR 037	103237	Load Buffer from B Register
OME 037	103037	Load Buffer from Memory
INA 037	102137	Read Buffer into A Register
INB 037	102237	Read Buffer into B Register
IME 037	102037	Read Buffer into Memory
CIA 037	102537	Read Buffer into Cleared A Register
CIB 037	102637	Read Buffer into Cleared B Register
C. Sense		
SEN 0537	101537	Sense Buffer Ready

Figure 13-4. High-Speed Paper Tape I/O Instruction Codes

Seven-Track Magnetic Tape Equipment

The seven-track magnetic tape system consists of up to four magnetic tape transports and a magnetic tape controller (Model 620-31A, B, C). The system can read and record a magnetic tape that is IBM 2400-compatible for seven-track systems.

The magnetic tape controller provides a buffered interface between the Varian 620/f I/O bus and a seven-track magnetic tape transport. The controller accommodates up to four transports of the Peripheral Equipment Corporation 6000 series. However, only one tape transport can be selected for use at any one time.

The tape controller comprises two circuit cards and contains all read/write data registers and timing and control logic required to control one tape transport.

Computer control of the magnetic tape system is accomplished through the I/O bus. The controller can also be operated under direction of the BIC.

When more than one tape transport is used with the controller, the transports are connected to the controller in party-line configuration. However, only one transport may be operated at a time. Transport selection is program-controllable.

Organization	Consists of instruction decode and storage sections, sense, read/write data control and write motion control, read motion control, read/write data storage and error checking sections, clock, and drivers and receivers.
I/O Capability	Eight external control, eight transfer, eight sense, and four transport select instructions
Control Capability	Four tape transports any one of which may be selected for connection to the controller. System reset automatically selects transport 1
Data Word	Buffering provided for two 16-bit words, each word containing up to three bytes
Error Checking	LRC and CRC characters are generated during read. Error correction is not provided.
Input Power	+5V at amperes
Logic Levels	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to +0.5V dc
Size	Two 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket circuit cards
Interconnection	Each card interfaces with the computer and BIC option via a 122-terminal connector. Each card interfaces with the tape transport via two 44-terminal connectors.

Figure 13-5. Seven-Track Controller Specification

Mnemonic	Octal	Function
A. External Control		
EXC 010	100010	Read One Record Binary
EXC 0110	100110	Read One Record BCD
EXC 0210	100210	Write One Record Binary
EXC 0310	100310	Write One Record BCD
EXC 0410	100410	Write File Mark
EXC 0510	100510	Forward One Record
EXC 0610	100610	Backspace One Record
EXC 0710	100710	Rewind
B. Transfer		
ØAR 010	103110	Load Buffer from A Register
ØBR 010	103210	Load Buffer from B Register
ØME 010	103010	Load Buffer from Memory
INA 010	102110	Read Buffer into A Register
INB 010	102210	Read Buffer into B Register
IME 010	102010	Read Buffer into Memory
CIA 010	102510	Read Buffer into Cleared A Register
CIB 010	102610	Read Buffer into Cleared B Register
C. Sense		
SEN 010	101010	Sense Parity Error
SEN 0110	101110	Sense Buffer Ready
SEN 0210	101210	Sense MTU Ready
SEN 0310	101310	Sense File Mark
SEN 0410	101410	Sense Old Length Record/High Density
SEN 0510	101510	Sense End of Tape
SEN 0610	101610	Sense Beginning of Tape
SEN 0710	101710	Sense Rewinding
D. Transport Select		
EXC2 0110	104110	Select MTU #1
EXC2 0210	104210	Select MTU #2
EXC2 0310	104310	Select MTU #3
EXC2 0410	104410	Select MTU #4

Figure 13-6. Seven-Track Magnetic Tape Unit Instruction Codes

Nine-Track Magnetic Tape Equipment

The nine-track magnetic tape system consists of up to four Peripheral Equipment Corporation Series 6000 tape transports and one magnetic tape controller (MTC), Model 620-30, for processing IBM 2400-compatible tapes.

The MTC provides a buffered interface between the Varian 620/f I/O bus and the tape transport. The MTC accommodates up to four tape transports, but only one of these is in use at any given time.

The MTC is on two wire-wrapped socket cards which can be installed in an expansion chassis. It contains all read/write registers and logic circuitry for the tape transport control.

Computer control of the magnetic tape system is accomplished through the I/O bus. The controller can also be operated under direction of the BIC.

If the system contains more than one tape transport, the transports are connected to the MTC in party-line configuration. The program controls the selection of the one transport that can operate at any given time.

Organization	Consists of a clock, drivers, receivers, and the following logic sections: instruction decoding, instruction storage, sense, read/write motion control, read/write data control, read/write data storage, and error checking.
I/O Capability	Six external control, eight transfer, eight sense, and four transport select instructions
Control Capability	Can select one of up to four tape transports at any given time. Resetting the system automatically selects tape transport 1.
Data Word	Buffering is provided for two 16-bit words, each containing two 8-bit bytes.
Error Checking	During writing, cyclic redundancy check (CRC) characters and longitudinal redundancy check (LRC) characters are written for each tape record. During reading, these characters are regenerated and compared with those read. The LRC includes a parity check. Error correction is not provided.
Logic Levels	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to +0.5V dc
Size	Two 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket circuit cards
Interconnection	Each card interfaces with the computer and BIC via a 122-terminal connector. Each card interfaces with tape transport via two 44-terminal connectors.
Input Power	+5V dc at 3 amperes nominal
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-7. Nine-Track Magnetic Tape Controller Specification

Mnemonic	Octal	Functional Description
A. External Control		
EXC 010	100010	Read One Record Binary
EXC 0210	100210	Write One Record Binary
EXC 0410	100410	Write File Mark
EXC 0510	100510	Forward One Record
EXC 0610	100610	Backspace One Record
EXC 0710	100710	Rewind
B. Transfer		
OME 010	103010	Output Memory to Magnetic Tape Buffer
OAR 0110	103110	Output A Reg to Magnetic Tape Buffer
OBR 0210	103210	Output B Reg to Magnetic Tape Buffer
IME 010	102010	Input Magnetic Tape Buffer to Memory
INA 0110	102110	Input Magnetic Tape Buffer to A Register
INB 0210	102210	Input Magnetic Tape Buffer to B Register
CIA 0510	102510	Input Magnetic Tape Buffer to A Register Cleared
CIB 0610	102610	Input Magnetic Tape Buffer to B Register Cleared
C. Sense		
SEN 010	101010	Sense Tape Error
SEN 0110	101110	Sense Buffer Ready
SEN 0210	101210	Sense Tape Unit Ready
SEN 0310	101310	Sense File Mark
SEN 0410	101410	Sense Odd Length Record/Sense High Density
SEN 0510	101510	Sense End of Tape
SEN 0610	101610	Sense Beginning of Tape
SEN 0710	101710	Sense Rewinding
D. Transport Select		
EXC2 0110	104110	Select Tape Drive No. 1
EXC2 0210	104210	Select Tape Drive No. 2
EXC2 0310	104310	Select Tape Drive No. 3
EXC2 0410	104410	Select Tape Drive No. 4

Figure 13-8. 9-Track Magnetic Tape Unit Instruction Codes

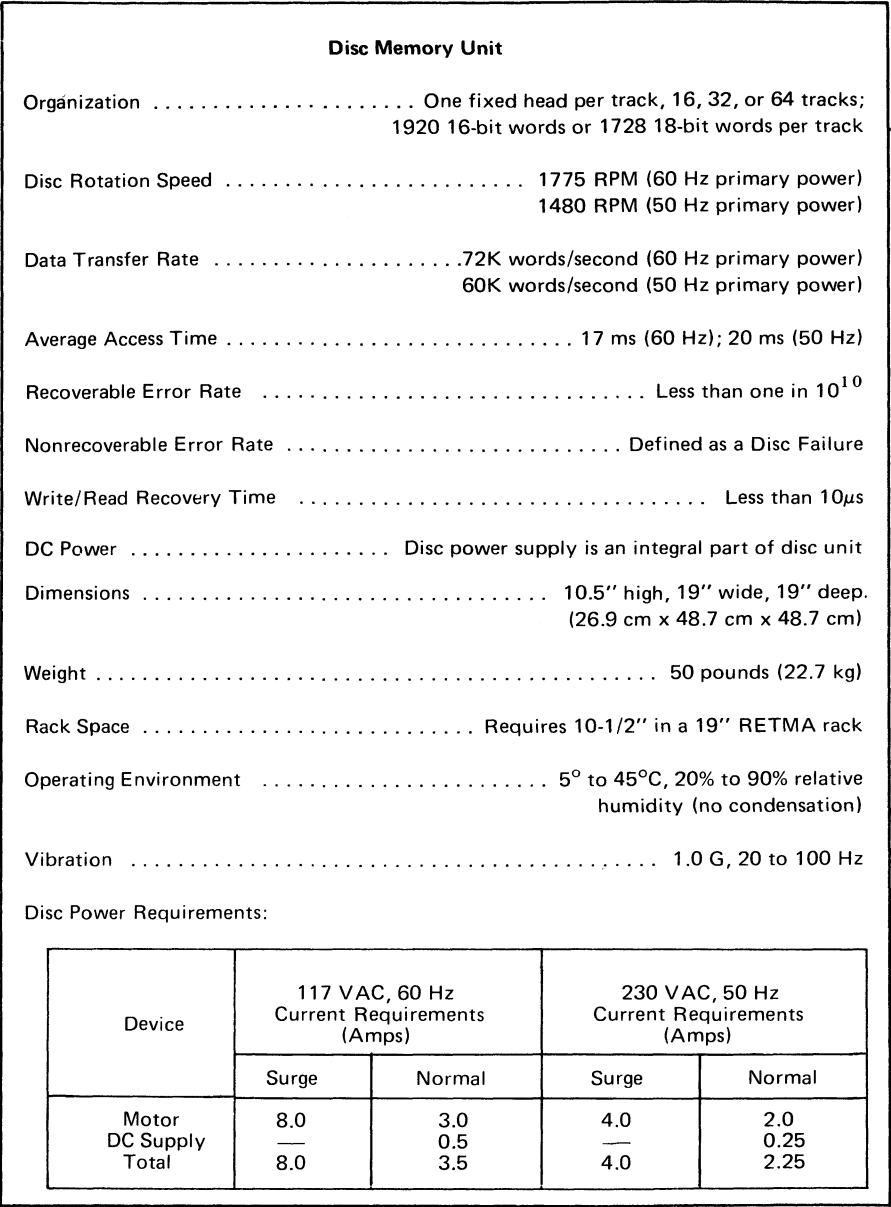


Figure 13-9. Disc Memory Specifications, Models 620-38 A, B, C (1 of 2)

Controller	
Control Capability	One disc unit
Operational Modes	Read and Write under BIC control
Construction	Logic mounted on one 7-3/4" by 12-11/16" wire-wrap board (19.8 cm by 33 cm)
Power Required	+5 Vdc @ 10 Watts nominal (supplied by 620/i power supply)
Operating Environment	5° to 45°C, 10% to 90% relative humidity (no condensation)
Interface Cable	10 feet long (3.07 meters); connects from controller card edge to disc unit

Figure 13-9. Disc Memory Specifications, Models 620-38 A, B, C (2 of 2)

Type	Single-disc, cartridge-type disc storage unit
Cartridge Type	IBM 2315 or VDM-approved equivalent
File Protection	Front Panel Switch Selection
Track Spacing	0.010 inches
Transfer Rate	42,000 16-bit words/sec 38,000 18-bit words/sec
Capacity (per Cartridge)	585,000 16-bit words; 520,000 18-bit words
Rotation Speed	1500 rpm
Track Access Time:	
Maximum	477 milliseconds
Minimum	2.35 milliseconds
Settle Time	20 milliseconds

Figure 13-10. Disc Memory Specifications, Model 620-39 (1 of 2)

Mean Time Between Failures (MTBF)	Greater than 1700 hours
Power Requirements	Supplied by disc power supply
Dimensions	19" wide, 17-1/2" high, 28-1/2" deep (48.6 x 44.9 x 73.1 cm)
Weight	Approximately 95 pounds (43.2 kg)
Operating Environment	19° to 32°C; 20 to 80% relative humidity (no condensation)
Disc Power Supply	
Capability	Provides all dc and ac power required by two disc drive units
Input Power Requirements	115VAC 60 Hz, or 230VAC 50 Hz
Dimensions	8-3/4" high, 19" wide, 23.6" deep (22.4 x 48.7 x 60.5 cm)
Operating Environment	19° to 32°C; 20 to 80% relative humidity (no condensation)
Disc Controller	
Control Capability	Provides control of two disc units
Construction	One wire-wrap circuit board; 7-3/4" by 12-11/16" (19.8 cm x 31.2 cm approx.)
Mounting	Requires three card slots in a 620/i-01 Expansion Chassis
Operating Power	+5Vdc @ 1.5A
Operating Environment	5° to 45° C, 0% to 90% relative humidity (no condensation)

Figure 13-10. Disc Memory Specifications, Model 620-39 (2 of 2)

Mnemonic	Machine Code	Description
External Control (Mode Selection)		
EXC 014	100014	Select Read Mode and connect controller to BIC
EXC 114	100114	Select Write Mode and connect controller to BIC
Sense (Status Indicators)		
SEN 014	101014	Disc Failure
SEN 114	101114	Disc ready to execute a mode select instruction
SEN 214	101214	Disc is busy with block transfer operation
SEN 314	101314	Illegal address selected
SEN 414	101414	Read parity error detected
SEN 514	101514	Track timing error
SEN 614	101614	Data transfer timing error
Transfer		
OME 014	103014	Output Memory as Starting Sector Disc Address
OAR 114	103114	Output A Register as Starting Sector Disc Address
OBR 214	103214	Output B Register as Starting Sector Disc Address

Figure 13-11. Disc I/O Instruction Codes, Models 620-38 A, B and C

Mnemonic	Machine Code	Description
External Control		
EXC 014	100014	Select Read Mode and Connect Controller to BIC
EXC 114	100114	Select Write Mode and Connect Controller to BIC
EXC 214	100214	Select Disc A For Move
EXC 314	100314	Select Disc B For Move
EXC 414	100414	Select Disc A For Data
EXC 514	100514	Select Disc B For Data
EXC 614	100614	Enable Interrupt
EXC 714	100714	Initialize Discs A & B
Transfer		
OME 14	103014	Output Memory as Starting Disc Sector Address or Move Information
OAR 14	103114	Output A Register as Starting Disc Sector Address or Move Information
OBR 14	103214	Output B Register as Starting Disc Sector Address or Move Information
Sense		
SEN 014	101014	Disc or Controller Failure
SEN 114	101114	File is Protected
SEN 214	101214	Disc is Busy
SEN 314	101314	Disc A Head at Track 0 (Home Position)
SEN 414	101414	Disc B Head at Track 0 (Home Position)
SEN 514	101514	Move Completed (Disc A or B)
SEN 614	101614	Read Parity Error/Write Overwrite
SEN 714	101714	Position Error

Figure 13-12. Disc I/O Instruction Codes, Model 620-39

Drum Memories

The drum memory system (Models 620-44 through -49) consists of a drum controller circuit card, a drum memory unit, and a dc power supply.

The drum memory system is field-expandable to accommodate increased system storage requirements. The smaller of the two drums is expandable in increments of 16 tracks up to a maximum of 128 tracks. The larger model is expandable in increments of 64 tracks up to a maximum of 512 tracks. Each track has 1,920 words (16 bits of storage). This provides a maximum storage capacity of 983,040 16-bit words in a 512 data track system. For a 128 data track system, the storage is 245,760 sixteen-bit words. A computer word stored on the drum unit consists of 16 data bits and one parity bit.

The controller circuit card contains all data registers and timing and control logic required to control one drum memory unit. The drum memory unit contains a single rotating drum, mechanical drive assemblies,

and read/write control logic. The dc power supply provides required operating voltages.

Control of the drum memory system by the Varian 620/f computer is accomplished under direction of the BIC. The BIC is a prerequisite of the drum memory system. Data transfer between the drum controller and drum memory unit is accomplished via the drum cable.

The drum controller performs the following functions:

- a. Controls bit serial data transfer between the controller and drum memory unit.
- b. Monitors and detects parity errors during read-from-drum operations.
- c. Controls mode of operation and provides all interface control between the computer and the drum memory unit.

Instruction codes for the drum memory are identical to those for the Model 620-38 disc (Figure 13-11).

Organization	Consists of a timing and control section, buffer register, shift register, address register/counter, location counter, comparator, and interface drivers and receivers.
Control Capability	One drum memory unit with up to 983,040 17-bit words (16 data bits plus odd parity).
I/O Capability	Two external control, three transfer, and seven sense instructions
Logic Levels	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to +0.5V dc
Operational Modes	Read block transfer; write block transfer
Word Transfer Rate:	
60-Hz Primary Power	106,000 sixteen-bit words per second
50-Hz Primary Power	88,500 sixteen-bit words per second
Priority Assignment	BIC-dependent
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket circuit board
Interconnection	Interfaces with the computer and BIC via a 122-terminal connector. Interfaces with drum unit via two 44-terminal connectors.
Input Power	+5V dc at 2 amperes
Operational Environment	5 to 45 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-13. Drum Memory Controller Specifications

Line Printer Equipment

The Model 620-77 buffered line printer represents an operational, self-contained subsystem consisting of a high-speed line printer and an interface controller.

This option offers high performance and printing quality to meet user requirements in a wide range of on-line applications. The line printer can also be used for off-line activities by incorporating the required interface control logic.

Significant features and characteristics of the line printer are:

- a. Segmented buffered line storage of up to a 132 six-bit character capacity
- b. Programmed line space control using standard TTY paper tape format.
- c. 245 to 1100 line-per-minute printing speeds providing 63 graphic and one blank (ASCII) character codes.
- d. High reliability and printout quality as a result of friction-free, one-piece hammer construction.

Printer	
Character Data:	
Format	Standard ASCII
Codes	63 printable in one blank
Characters Per Line	Up to 132
Horizontal Spacing	Ten characters per inch
Vertical Spacing	Six characters per inch
Paper:	
Dimensions	Standard fanfold, edge-punched 4 to 19 inches (8.1 to 48.1 cm) wide with 22 inches (55.7 cm) between folds
Type	Single copy, 15-pound bond minimum-weight multicopy, up to six parts, 12-pound bond with shot carbon
Ribbon:	
Dimensions	Vertically fed roll type 14 inches (33.4 cm) wide, 20 yards (18m) long
Cabinet:	
Dimensions	46 inches high, 48.5 inches wide, 25 inches deep
Weight	Approximately 575 lbs.
Dynamic Characteristics:	
Printing Speed	245 lines per minute, 132 columns 290 lines per minute, 120 columns 356 lines per minute, 96 columns 460 lines per minute, 72 columns 650 lines per minute, 48 columns 1,110 lines per minute, 24 columns
Drum Rotation	1760 rpm
Primary Voltage	110 to 130V ac
Range	Single phase 60 Hz
Power Requirements	500 watts
Equipment:	
Operating Temperature	50 to 110°F
Relative Humidity	30 to 90 percent without condensation
Controller	
Organization	Consists of timing and control-select/deselect logic, word buffer, drivers, and receivers
Control Capability	One printer, six-bit data words

Figure 13-14. Line Printer Specifications (1 of 2)

Output Capability	Two external control, three transfer, and two sense instructions
Logic Levels:	
I/O and B Cables	Negative Logic True: 0.0 to 0.5V dc False: +2.5 to +3.7V dc
Internal	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to 0.5V dc
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket circuit card
Interconnection	Interfaces with the computer via a 122-terminal connector; interfaces with line printer via two 44-terminal connectors
Input Power	+5V dc at 500 mA
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-14. Line Printer Specifications (2 of 2)

Mnemonic	Octal	Functional Description
A. External Control		
EXC 035	100035	Initialize Printer
EXC 0135	100135	Connect Printer to BIC
B. Sense		
SEN 035	101035	Sense line printer ready
SEN 0135	101135	Sense character buffer ready
C. Transfer		
OME 035	103035	Transfer memory to printer buffer
OAR 035	103135	Transfer A register to printer buffer
OBR 035	103235	Transfer B register to printer buffer

Figure 13-15. Line Printer System Controller Instructions

Oscilloscope Display System

The Model 620-73 oscilloscope display system provides visual display of the computer output. The system includes a Tektronix model 602 oscilloscope with a 5-inch (8.7 cm) rectangular cathode ray tube (CRT), a controller, reference power supply, and interconnecting cables.

The oscilloscope controller converts digital data to analog values which drive the horizontal and vertical deflection plates of the oscilloscope. The oscilloscope is specially designed for X-Y presentation. One of the D/A converters in the controller drives the oscilloscope X axis; the other, the Y axis. The Z channel input turns the CRT beam on and off.

The display system is directly under program control. The display is programmed by outputting data to one of the two A/D converters. Portions of the display can be inhibited by the program.

Controller

Organization	Contains decoding logic, two D/A converters, Z axis control circuitry, and voltage regulator
I/O Capability	Four external control and six transfer instructions
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) wirewrapped-circuit board
Interconnection	Interfaces with the computer via backplane wiring panel; interfaces with the power supply via the backplane connector; X, Y, and Z axis signals route to the oscilloscope via three cables that card-edge-connect to J1
Input Power	+22V dc at 650 mA; -22V dc at 350 mA
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

D/A Converters

Type	12-bit; bipolar output
Accuracy	Better than 0.1 percent
Output	Single-ended, ± 10 volts full scale at 10 mA, short circuit
Settling Time	10 microseconds maximum
Conversion Time	100 Hz

Oscilloscope

Type	Portable monitor unit with self-contained power supply; 5-inch (8.7 cm) rectangular CRT
Inputs	X, Y, and Z axis signals
Size	6 inches (15.2 cm) high, 8.5 inches (21.5 cm) wide, 17.5 inches (44.3 cm) deep
Input Power	90 to 136V ac or 180 to 240V ac, 50 or 60 Hz
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-16. Oscilloscope Display System Specifications

Mnemonic	Octal	Functional Description
External Control		
EXC 056	100056	Initialize X-axis
EXC 057	100057	Initialize Y-axis
EXC 256	100256	Turn On CRT
EXC 356	100356	Turn Off CRT
Transfer		
OAR 056	103156	Transfer A Register to X-axis
OBR 056	103256	Transfer B Register to X-axis
OME 056	103056	Output Memory to X-axis
OAR 057	103157	Transfer A Register to Y-axis
OBR 057	103257	Transfer B Register to Y-axis
OME 057	103057	Output Memory to Y-axis

Figure 13-17. Oscilloscope Display Control Instructions

Punched Card Equipment

Punched Card Reader

The Model 620 -25 card reader system (CRS) is a peripheral option that reads data from 80-column (51-column optional) punched cards and transfers the data to a Varian 620/f computer. The CRS consists of a Bridge Data Products Model 8000 card reader and a card reader controller.

The controller is constructed with 28 integrated circuit chips and discrete component assemblies mounted on a standard wired-socket card. The controller can be mounted in any universal card slot in the mainframe or expansion chassis.

The card reader reads the information from punched cards and provides the data to the controller. The card reader controller provides a nonbuffered interface between the card reader and the CPU. It also provides the timing and logic circuits to effect the transfers. The controller can transfer data to the CPU under direct program control or under supervision of a BIC.

Card Punch

The Model 620-27 card punch controller controls data transfer from the computer to the card punch. The controller permits punch operation under CPU control and, optionally, BIC control.

The data buffer register in the controller stores the 12-bit data words from the CPU. The control section synchronizes and controls punch operation. A data present strobe to the punch indicates that correct data are on the data lines.

Under program control, the controller senses the punch (ready or not busy) and transfers data to it. Under BIC control, the controller requests data and the BIC controls the transfer from the specified memory addresses. Transfer continues until terminated by the BIC.

punched card

Organization	Contains input receivers and output drivers for I/O bus and BIC interface operations and sequence control logic to transfer data from the card reader to the computer
Capability	Bridge Data Products Model 8000 (620-25)
Operating Speed	With Model 8000: 300 cards per minute
Character Length	12 bits, LSB D100, MSB D111
Feed Type	Standard 80-column cards (51 optional) on a per-column basis (end feed)
Transfer Rate	Maximum rate between the 620/L and the controller: 1,050 characters per second
Logic Levels	Positive Logic True: +2.4 to +5.5V dc False: 0.0 to +0.5V dc
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket circuit card
Interconnection	Interfaces with the computer and BIC via a 122-terminal connector; interfaces with the card reader via two 44-terminal connectors
Input Power	+5V dc at 280 mA
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-18. Card-Reader Controller Specifications

Organization	Contains the punch output data buffer, line drivers/receivers for I/O and punch cabling, and a control section
Priority Assignment	BIC-dependent
Timing:	
Data Present Strobe	Typically 15 microseconds
Card Punch Ready	False for approximately 12 microseconds while data buffer stores data
Character Length	12 bits
Feed Type	Punches standard 80-column cards (51 optional) on a per-column basis
Transfer Rate	260 characters per second
Logic Levels	Positive Logic
	True: +2.4 to +5.5V dc
	False: 0.0 to +0.5V dc
Input Power	-5V dc at 485 mA
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket board
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-19. Card Punch Controller Specification

Mnemonic	Octal	Function
A. External Control		
EXC 030	100030	Initialize
EXC 0230	100230	Read One Card
*EXC 0630	100630	Step Read One Character
B. Transfer		
INA 030	102130	Transfer to A Register
INB 030	102230	Transfer to B Register
IME 030	102030	Transfer to Memory
CIA 030	102530	Transfer to A Register Cleared
CIB 030	102630	Transfer to B Register Cleared
C. Sense		
SEN 0130	101130	Sense Character Ready
**SEN 0230	101230	Sense Reader Error
***SEN 0330	101330	Sense Hopper Empty
SEN 0630	101630	Sense Reader Ready

Figure 13-20. Card Reader Instruction Codes

*Available only with ERD model

**Sense reader not error on ERD

***Not available in ERD model

Digital Plotter

The Model 620-72 digital plotter system consists of a Calcomp 565 digital plotter and plotter controller card. The plotter produces high-quality, ink-on-paper graphic presentations of computer output.

The plotter controller provides a fully buffered interface to permit operation of the plotter under program control or under the direction of BIC.

The plotter uses digital commands from the computer to produce the plot or drawing on a 12-inch wide roll of paper. These commands actuate step motors to produce incremental movement of the pen with respect to the paper. One step motor controls movement in two directions along the X axis, and a separate motor controls movement on the Y axis. X-axis movement is accomplished by rotating the paper drum in either the +X or -X direction. The pen is moved to the left or right to effect movement on the Y axis. Composite commands can be given to move the pen on two axes simultaneously. This results in commands to move the plotter pen in any of eight directions. Model 565 operates at up to 300 increments per second, with three optional increment sizes: 0.005 inches, 0.010 inches, and 0.1 millimeters. The increment size is specified when ordering.

Plotter	
General	Digital commands actuate step motors to produce plot
Increment Sizes (selected when ordering)	0.005 inch (0.01 cm); 0.010 inch (0.03 cm); 0.1 mm
Speed	300 increments per second (maximum)
Plot Dimensions:	
X Axis	11 inches (27.8 cm)
Y Axis	120 feet (36.4m)
Paper	12 inches wide by 120 feet long edge-punched roll; available in a wide variety of patterns printed on various stocks
Pen	Liquid ink or ballpoint pen
Power Requirements	105 to 125V ac, single phase, 50/60 Hz, 1.5A (maximum)
Size	98 inches (248 cm) high, 18 inches (45.5 cm) wide, and 14.7 inches (35.2 cm) deep
Operational Environment	0 to 45 degrees C, 0 to 96 percent relative humidity without condensation
Controller	
Organization	Contains interface circuitry, control and timing logic, and six-bit command buffer
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) etched-circuit card
Input Power	+5V dc at 380 mA; -5V dc at 42 mA; -12V dc at 15 mA
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-21. Digital Plotter Specifications

Mnemonic	Octal	Functional Description
A. External Control		
EXC 032	100032	BIC to DPC Enable
B. Transfer		
OME 032	103032	Transfer Memory to Buffer
OAR 032	103132	Transfer A Register to Buffer
OBR 032	103232	Transfer B Register to Buffer
C. Sense		
SEN 0132	101132	Sense Buffer Ready

Figure 13-22. Digital Plotter Controller Instructions

Buffered I/O Controller

The Model 620-80 buffered I/O controller provides a self-contained, programmable hardware interface for general-purpose data-handling.

The input and output buffer registers provide parallel word data communications between the computer I/O bus and an external device. In addition to data handling, the output buffer register can be programmed to output discrete control bits to an external device.

The buffered I/O controller uses a customer-fabricated U cable, up to 20 feet (6m) long, for communication with external devices.

Organization	Contains operation and function decoding logic, input and output buffer registers, eight sense input gates and eight variable-pulsewidth control gates, and interface drivers and receivers
Capability	Provides buffered data transmission to/from external devices and the 620/f
Sensing Line Input Current Load	Nominally 7 mA source at 0 volt
Buffered Output Register Current Load	Nominally 36 mA source at 0 volt
Control Pulse Output Current Drive	Up to 65 mA sink at 0 volt
Logic Levels	Positive Logic True: +2.5 to +5.5V dc False: 0.0 to 0.5V dc
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-socket card
Interconnection	Interfaces with the computer via a 122-terminal connector; interfaces with external devices via two 44-terminal connectors
Input Power	+5V dc at 1.8 amperes
Operational Environment	0 to 50 degrees C, 10 to 90 percent relative humidity without condensation

Figure 13-23. Buffered I/O Controller Specifications

Mnemonic	Octal	Functional Description
A. External Control		
EXC 0X6Z*	100X6Z	Output control pulse on line selected by X from controller addressed by Z.
B. Sense		
SEN 0X6Z	101X6Z	Test state of line selected by X from controller addressed by Z.
C. Input Data		
IME 06Z	10206Z	Read input buffer of controller addressed by Z into memory.
INA 016Z	10216Z	Read input buffer of controller addressed by Z into B register.
INB 026Z	10226Z	Read input buffer of controller addressed by Z into B register.
CIA 056Z	10256Z	Clear A register and read controller input buffer addressed by Z.
CIB 066Z	10266Z	Clear B register and read controller input buffer addressed by Z.
D. Output Data		
OME 06Z	10306Z	Load output buffer of controller addressed by Z from memory.
OAR 016Z	10316Z	Load output buffer of controller addressed by Z from A register.
OBR 026Z	10326Z	Load output buffer of controller addressed by Z from B register.
*6Z = Device address (60-67). Determined on individual system basis by wiring on backplane of peripheral expansion chassis.		
* X = Discrete control/sense line (0-7).		

Figure 13-24. Buffered I/O Controller Instructions

Relay I/O Module

The Model 620-83 relay I/O option provides a general-purpose, relay-buffered data link between special peripherals and the 620/f. The option has the capability of 16 relay-buffered inputs (620-83-2), 16 mercury-wetted relay contact outputs (620-83-1), or combined input/output (620-83-3). The relay input and output options are each packaged on one 7.75-by-12-inch (19.7 x 30.3 cm) socket board. Printed circuit etch is used for the relay portion of the board.

In the Model 620-83-2, input relay contacts are activated by voltages from the user's equipment through the 12V dc input relay coil (10 volts at 6.5 mA minimum). Series resistors for coil input voltages greater than 12 volts can be installed. An energized input relay coil closes a contact that is gated into a flip-flop. This 16-bit flip-flop register can be accessed by the Varian 620/f with one of five data input commands. The flip-flop register can be cleared with an external control command or by system reset. The flip-flops remain set after an initial relay input until they are cleared.

The Model 620-83-1 relay-output option allows isolated parallel data transfer of a 16-bit word from the 620/f via mercury-wetted relay contacts to the user's equipment. A 16-bit word is clocked into a flip-flop register by any of three data transfer out commands. The register drives 16 discrete circuits that, in turn, drive the 12-volt relay coils closing the contacts.

The relays can be cleared by an external control instruction, transferring all-zeros data out, or by system reset. The relay contacts remain closed until the flip-flops are cleared.

I/O Capability	Two external control, one sense, and eight data transfer instructions
Relay Type:	
Output	Mercury-wetted reed
Input	Dry reed
Contact Rating:	
Output	50 volt-amperes resistive 3 amperes or 400 volts maximum
Input	12 volt-amperes resistive 1/2 ampere or 200 volts maximum
Contact Resistance:	
Output	0.05 ohm average
Input	0.2 ohm average
Capacitance:	
Output	1 pF plus 0.01 μ F external
Input	Less than 1 pF
Operating Time:	
Output	2 milliseconds average
Input	1 millisecond average including bounce
Release Time:	
Output	2 millisecond average
Input	1 millisecond average
Coil Power Consumption:	
Output	500 mW average
Input	100 mW average
Drive Requirement:	
Output	3 TTL load (through an amplifier)
Input	10 volts at 6.5 mA minimum
Life at Rated Load:	
Output	25×10^6 operations
Input	100×10^6 operations
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) socket board
Operational Environment	0 to 50 degrees C, 0 to 90 percent relative humidity without condensation

Figure 13-25. Relay I/O Module Specifications

Mnemonic	Octal	Functional Description
A. External Control		
EXC 0DA	1000DA	Clear All Outputs. Causes all 16 output contacts to open.
EXC 1DA	1001DA	Clear All Inputs. Returns all input bits that are not being set by contact closure to zero.
B. Sense		
SEN 0DA	1010DA	Sense Contact Closed. This command is available as an option. A specified contact closure will cause a jump to the jump address to occur.
C. Transfer In		
INA 0DA	1021DA	Input to A register. Input relay buffered input data on module to A register.
CIA 0DA	1025DA	Clear and Input to A register. Input relay buffered input data on module to A register cleared.
INB 0DA	1022DA	Input to B register. Input relay buffered input data on module to B register.
CIB 0DA	1026DA	Clear and Input to B register. Input relay buffered input data on module to B register cleared.
IME 0DA, ADDR	1020DA	Input to Memory. Input relay buffered input data on module to memory.

Figure 13-26. Relay I/O Module Control Instructions

Digital I/O Controller

The Model 620-81 digital I/O controller (DIOC) provides a programmed link between an external device and the computer. There are eight separate control and sensing lines to permit the user to initialize, implement iterative control sequences, synchronize otherwise asynchronous external devices, and monitor the operational state of an external device.

The DIOC operates entirely under program control. The function code defines one of eight individual control or sensing lines. The DIOC responds to an EXC command by placing a pulse on the selected output control line. Similarly, it responds to a SEN command by testing the operational state of an external device via the true or false level applied to the selected sense-response line.

The DIOC uses a customer-fabricated U cable, up to 20 feet (6m) long, for communication to external devices.

Organization	Contains address and function decode logic, interface drivers and receivers, eight nonbuffered control pulse output gates, and eight nonbuffered sense response input gates
Controller Circuitry	Eight control pulse gates and eight sense line gates
Expansion Capability	Standard plug-in expansion of up to 64 control and sense lines
Control Pulse Output Current Drive	48 mA sink at 0 volt
Sense Response Input Current Load	6.5 mA source at 0 volt
Control Pulsethwidth	200 nanoseconds
Control Pulse	100 nanoseconds maximum
Transition Time	5 nanoseconds minimum
Logic Levels	Positive Logic True: +2.5 to +5.5V dc False: 0.0 to +0.5V dc
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) etched-circuit card
Interconnection	Interfaces with the computer via a 122-terminal connector; interfaces with external devices via two 44-terminal connectors
Input Power	+5V dc at 250 mA
Operational Environment	0 to 50 degrees C, 0 to 90 percent relative humidity without condensation

Figure 13-27. Digital I/O Controller Specifications

Mnemonic	Octal	Functional Description
A. External Control		
EXC	100XZZ*	Select device address of ZZ and initiate a control pulse on line X.
B. Sense		
SEN	101XZZ	Select device address of ZZ and test logical state of sense response line X.
*ZZ = Device address 60 ₈ to 67 ₈		
* X = Control or sense line 0 through 7		

Figure 13-28. Digital I/O Controller Instructions

Analog Input Modules

The Model 620-850 (single ended) and Model 620-851 (differential) are low-cost, general purpose analog-input modules designed for use on the Varian 620/f computer. The modules are completely self-contained and are plug-in compatible with the Varian 620/f computer mainframe or expansion frame. The modules provides the following significant features:

- Analog-to-digital converter
- Sample and hold amplifier
- Programmable timer
- I/O and BIC interface
- Multiplexer
- Wired for Priority Interrupt Module (PIM) option

Each module includes an analog-to-digital converter with 13 bit resolution and 55 KHz throughput rate, sample and hold amplifier, programmable timer and multiplexer with 16 differential or 32 single-ended channels expandable to 256 differential or 512 single-ended channels.

The analog-to-digital converter provides a digital output, corresponding to an analog voltage input. The analog-to-digital conversion can be initiated by the computer, by the programmable timer or by an external pulse.

The programmable timer provides an internal and external control capability. Under internal control, the timer sets the timing interval through a data word which it receives from the computer. The timer decrements the data word at 1 microsecond per count until the zero state is reached. The timer then emits a 100 ns pulse, restores the

original data word and again initiates the cycle.

Under external control, the timer initiates the timing cycle upon receipt of an external signal. As in internal control, the timer decrements the data word until the zero state is reached. Then, the timer will stop and wait for an external signal before the next cycle is initiated. During the wait period, a new timing interval can be sent by the computer.

The multiplexer can operate in either of two modes:

- Scan mode
- Program mode

The Program Mode allows the multiplexer address to occur at variable intervals determined by the computer. The selected channel is read and the data can be converted to its digital representation.

The end-of-conversion signal can generate an interrupt via the optional Priority Interrupt Module (PIM), which can start a program to select the next channel address from a "channel" table.

With the Buffer Interface Controller option, instead of generating an interrupt, the end-of-conversion signal initiates a trap-out sequence via a BIC connected to the multiplexer.

In the Scan Mode, channel addressing is selected sequentially. Each scan starts automatically with the first channel and a channel advance signal, increments the multiplexer to the next higher channel. At the end of a scan cycle, the multiplexer is set to the first channel and an end-of-scan interrupt can be monitored by the computer.

Resolution	13 Binary bit
Analog Input (Full Scale)	± 10 volts
Conversion Accuracy	$\pm .012\%$ of 20 volts Full scale $\pm 1/2$ LSB
Conversion Time (13 bit)	14 μ seconds, maximum
Throughput Rate	55 KHz, maximum
Temperature Coefficient	$\pm 100 \mu\text{V}/^{\circ}\text{C}$, maximum including sample and hold
Warm-up Time	Essentially zero
Output Format	2's complement or offset Binary
External Start	1K ohms to +5 VDC. Lower to start. Must raise and relower to restart.
Power	+15 VDC, $\pm .1\%$, 75 ma -15 VDC, $\pm 1\%$, 75 ma -20 VDC, $\pm 2\%$, 2 ma +5 VDC, $\pm 5\%$, .75 amps.
Environment	
Operating Temperature	0 $^{\circ}$ C to 50 $^{\circ}$ C
Storage Temperature	-55 $^{\circ}$ C to 85 $^{\circ}$ C
Sample and Hold	
Gain and Accuracy	
Voltage Gain	+1
Accuracy	$\pm .01\%$
Gain Temp. Coeff.	$\pm 10 \text{ PPM}/^{\circ}\text{C}$
Track Mode	
Full Power Response (20V peak to peak one wave)	150 KHz
Slew Rate	10v/ μ s
Settling Time to ± 1 mV	3 μ seconds

Figure 13-29. Analog Input Module Specifications (1 of 3)

Input Characteristics

Signal Range	$\pm 10\text{V}$
Maximum Rating (without damage)	$\pm 30\text{V}$
Input Impedance	10K ohms
Common Mode Rejection	-80 db, 0 to 60 Hz
Offset Voltage	$\pm 1\text{ mV}$, maximum
vs Temperature	$\pm 50\text{ }\mu\text{V}/^{\circ}\text{C}$

Output Characteristics

Signal Range	$\pm 10\text{V}$
Noise, RMS, Wide Band (Hold Mode)	1 mV peak to peak
Decay Rate in Hold Mode	$\pm 10\text{mV/second}$
Feedthrough 20V step (Hold Mode)	-80 db

Switching Characteristics

Aperture time	100 NS, maximum
Offset pedestal	$\pm 10\text{ mV}$, maximum
Acquisition Time	3 $\mu\text{seconds}$

Programmable Timer

Clock Frequency	1.0 MHz $\pm 0.1\%$
Clock Drift	$\pm 100\text{ PPM}$ over 0°C to 50°C
Clock Stability	$\pm 100\text{ PPM}$ in 3 years $\pm 10\text{ PPM}$ per week at 25°C
Resolution	16 Binary Bits (Computer E -Bus $2^0 - 2^{15}$)
Programmed PRF	1 MHz to 15.26 Hz
Pulse Output	100 Nz, $I_{\text{sink}} \leq 100\text{ ma}$ $C_L \leq 1000\text{ pf}$

Figure 13-29. Analog Input Module Specifications (2 of 3)

	Multiplexer
Number of Channels	16 differential or 32 single-ended. Address signals available for external expansion to a total of 256 differential or 512 single-ended channels.
Input Voltage (Full Scale)	$\pm 10V$
Over Voltage Limits	$\pm 15V$ (without damage)
Over Voltage Recovery	13 μ seconds
Input Impedance	100 Meg ohms
Input Impedance (off)	10 Meg ohms
Crosstalk	-80 db, 0 to 1 KHz
Settling Time	13 μ seconds
Source Impedance for Rated Performance	1K, maximum
Operating Voltages	$\pm 15V$, -20V, +5V
Environment	
Operating Ambient	0°C to 50°C
Storage	-55°C to 85°C

Figure 13-29. Analog Input Module Specifications (3 of 3)

Analog Output Modules

Introduction

The analog output modules, Models 620-870 through 620-875, are low-cost, general purpose, digital-to-analog converter configurations designed as an optional feature for the Varian 620/f computer. The modules are completely self-contained on standard VDM printed circuit cards that plug-in the computer I/O slots in the mainframe or expansion frame. The family of modules offers a wide range of digital-to-analog converter configurations providing the following significant features:

- Multi digital-to-analog converter configurations with up to 14 bit resolution
- I/O and BIC interface
- Eight external control lines
- Eight sense lines
- Expandable up to 64 digital-to-analog converter configurations

An analog output module consists of one standard VDM printed circuit card, containing a digital-to-analog converter configuration, an I/O and BIC interface, eight external control (EXC) lines, and eight sense (SEN) lines. A system can be expanded to eight digital-to-analog converter configurations per device address and up to eight device addresses are available. Hence, a system can be expandable to 32 modules providing a maximum of 64 digital-to-analog converter configurations. The available digital-to-analog converter configurations are the following:

Model No.	Configuration
620-870	Single 10 Bit

620-871	Single 12 Bit
620-872	Single 14 Bit
620-870A	Dual, two 10 Bit
620-871A	Dual, two 12 Bit
620-872A	Dual, two 14 Bit
620-873	Mix, single 10 Bit and single 12 Bit
620-874	Mix, single 10 Bit and single 14 Bit
620-875	Mix, single 12 Bit and single 14 Bit

All digital-to-analog converter configurations can be initialized to zero volts output through the computer's system reset switch. An extended external control selects anyone of up to eight digital-to-analog converter configurations at a specific address. After the selection, the computer outputs data words to the digital-to-analog converter configuration without reselecting for each output word. A digital-to-analog converter configuration remains selected until another is selected or the system is re-initialized. Each digital-to-analog converter configuration has a storage register which holds data until update or System Reset.

A selected digital-to-analog converter configuration can be operated in conjunction with the Buffered Interlace Controller (BIC). In this direct transfer mode, a voltage waveform is obtained from the analog equivalent of a data set resident in computer memory storage. When a system contains more than one BIC, the desired configuration is enabled via an external control prior to selecting the BIC transfer.

analog output

Inputs			
Resolution	10 bit	12 bit	14 bit
Coding	Two's complement or offset binary. Input comes from computer registers or core memory.		
Outputs			
Voltage	± 10 volts full scale		
Current	± 5 ma max.	± 10 ma max.	± 10 ma max.
Loading			
Restive	500 ohms	1K ohms	1K ohms
Capacitive	2000 pf at specified settling time. Any value can be tolerated without amplifier oscillation.		
Short Circuit Protection	Output may be shorted to ground indefinitely with no damage to the module.		
Dynamic Characteristics			
Output Slew Rate	0.5 volts/μsec.	6 volts/μsec.	6 volts/μsec.
Noise	Less than ± 1/2 LSB peak maximum, due to activity on computer I/O bus.		
Settling Time	10 μsec. to stated accuracy		
Adjustments	FS, zero	FS, zero MSB	FS, zero 3 MSB
Accuracy	± 0.05%	± 0.012%	± 0.003%
Power Requirements	+5 VDC ± 5% 500 ma +15 VDC ± 0.1% 50 ma -15 VDC ± 0.1% 50 ma		
Digital Control Outputs			
Number	Eight		
Type	Open collector transistor. TI SN 75451 P. Dual Peripheral Driver. Sinks current when true.		

Figure 13-30. Analog Output Module Specifications (1 of 2)

Each output will sink 300 ma and stand-off +30 volts. Outputs are controlled by flip-flops which are set and reset by computer instructions.

Digital Sense Inputs

Number Eight

Type TTL logic levels. Ground true.

Open circuit inputs are held to +5 volt supply through 5.6K ohm resistors.

Physical Characteristics

Dimension	One printed circuit board with size 7-3/4" x 12" x 0.5". Plug-in compatible with standard 620 series hardware.
-----------	--

Connectors	One 122-terminal card-edge connector (mates with female connector on backplane.) Two 44-terminal card-edge connectors (mates with female connector on external device cable.)
------------	---

Environment

Warm-Up Time Essentially zero

Temperature Coefficient

Temperature Range	
Operating	0°C to 50°C
Storage	-55°C to 85°C

***NOTE:** Specifications apply to 10 bit, 12 bit, and 14 bit unless otherwise stated.

Figure 13-30. Analog Output Module Specifications (2 of 2)

Analog Power Supply Module

The Power Supply Module Model 620-88, is a low-cost, general purpose analog power supply. This module is an optional feature of the VDM 620 series computers, completely self-contained and designed to provide all the power requirements of the A/D Interface Products.

Electrical Characteristics				
Input Voltage	115/230 VAC $\pm$ 10% 47 Hz to 63 Hz			
Input Current	1.6 Amps RMS F.L.			
Output Voltages, Currents				
Standard	+5 VDC, 0.1% 5 amp. +15 VDC, 0.1% 1 amp. -15 VDC, $\pm$ 3% 1 amp.			
Optional	$\pm$ 20 VDC, $\pm$ 3% 250 ma. +24 VDC, $\pm$ 3% 500 ma.			
Output Performance				
Voltages	$\pm$ 15V	$\pm$ 20V	+5V	+24V
Load Regulation (N.L. to F.L.)	0.1%	0.1%	0.5%	1%
Line Regulation (Nominal Volt. $\pm$ 10%)	0.1%	0.15%	0.1%	1%
Temperature Coefficient	$\pm$ 0.015%	$\pm$ 0.025%	$\pm$ 0.015%	$\pm$ 0.025%
Ripple and Noise (N.L. to F.L.)	5 Millivolts P - P maximum			50 Mv max.
Transient Response (50% load change)	25 μ sec	25 μ sec	10 μ sec	50 μ sec

Figure 13-31. Analog Power Supply Module Specifications (1 of 2)

Short Circuit Protection	Stands any combination of outputs shorted to ground or to each other indefinitely.			
Overvoltage Protection	+5.0V only. Crowbar fires at +6.8V ± .4V			
Remote Sensing	+5V, +15V, −15V (tracks +15V)			
Operating Temperature	0° to 50°C without forced air cooling			
Current Limits				
Voltage	+5V	± 15V	± 20V	+24
i minimum	5.4A	1.3A	.26A	.51A
i short	1.1A	.15A	.35A	.6A

Mechanical Characteristics

Mounting	Rack mount on front or rear cabinet rails.
Dimensions	5-1/4" x 19" x 6"

Figure 13-31. Analog Power Supply Module Specifications

EXC 0YY	Connects ADC (YY) to BIC (YY) = 60-67
EXC 1YY	Starts ADC
EXC 2YY	Enables ADC to be started by internal lock or external pulse
EXC 3YY	Locks out external and lock starting pulses
SEN 0YY	Checks ADC conversion finished
SEN 1YY	Sense ADC timer lock for zero (ADC trigger)
SEN 2YY	Status of external pulse line
CIA & IAR 0YY CIB & IBR 0YY IME 0YY	Inputs from ADC YY
OAR 0YY OBR 0YY OME 0YY	Outputs data word to ADC timer buffer
EXC 0YY	Selects random mode, & enables connection to BIC
EXC 1YY	Select random mode, only
EXC 2YY	Select sequential mode
SEN 0YY	Sense end of scan
OAR 0YY OME 0YY OBR 0YY	Data from A, B, or memory to MOX to select channels to be scanned with 8-bit binary # for ck 0-255 in random mode, or ck # for last channel in scan in sequential mode.

Figure 13-31a. Analog Input Module Instructions

EXC XYY	(X is 0-7) Selects one of 8 DAC's on a controller to be connected to BIC. (YY) is device address 50-57.
SEN XYY	Sense for DAC ready for new data
OAR XYY OBR XYY OME XYY	Output to selected DAC (X) on controller (YY)

Figure 13-31b. Analog Output Module Instructions

Dataset Controllers

Dataset controllers interface with the computer and modems that are compatible with standard datasets. These controllers can operate in half- or full-duplex mode. They detect and establish input synchronization and switch to word mode for data transfers.

Data are transferred one of three ways:

- a. The controller can be connected to the BIC for operations requiring minimum program intervention. BIC can be connected for half-duplex operation to input or output data. In
- b. Full- or half-duplex operation can be completely controlled by the program.
- c. Inputting data to the PIM provides interrupt-controlled transfers.
- d. Receive/Transmit. The controller simultaneously transfers data in both directions.

full-duplex operation, I/O functions can be connected to the BIC and data flow can be controlled by the program.

Organization	Contains controller logic, timing circuits, and I/O drivers/receivers
I/O Capability	Eight external control, two transfer, and six sense instructions
Operational Modes	Full- or half-duplex
Transfer Rate	2,000 to 2,400 bits per second, fixed rate
Priority Assignment	Under program, interrupt, or BIC control
Logic Levels	Positive Logic True: +2.5 to +5.0V dc False: 0.0 to 0.5V dc
Input Power	+5V dc at 2.5 amperes
Size	One 7.75-by-12-inch (19.7 x 30.3 cm) etched-circuit card
Operational Environment	0 to 50 degrees C, 0 to 90 percent relative humidity without condensation

Figure 13-32. Dataset Controller Specifications (Model 620-65A)

dataset controllers

Mnemonic	Octal	Functional Description
A. External Control		
EXC 071	100071	Go to Search
EXC 0171	100171	Connect Write Buffer to BIC
EXC 0271	100271	Connect Read Buffer to BIC
EXC 0471	100471	Turn on Request to Send
EXC 0671	100671	Go to Character Format
B. Transfers		
IME 071	102071	Transfer Read Buffer to 8 LSB of Memory
INA 071	102171	Transfer Read Buffer to 8 LSB of A Reg.
INB 071	102271	Transfer Read Buffer to 8 LSB of B Reg.
CIA 071	102571	Transfer Read Buffer to 8 LSB of A Reg. cleared
CIB 071	102671	Transfer Read Buffer to 8 LSB of B Reg. cleared
OME 071	103071	Transfer Memory 8 LSB to Write Buffer
OAR 071	103171	Transfer A Register 8 LSB to Write Buffer
OBR 071	103271	Transfer B Register 8 LSB to Write Buffer
C. Sense		
SEN 0171	101171	Write Buffer Empty
SEN 0271	101271	Read Buffer Full
SEN 0371	101371	Carrier On
SEN 0471	101471	Clear to Send
Notes:		
1. All commands listed are used for 201A3 Dataset operation.		
2. EXC 571 is not necessary for 201B1 (true) full-duplex operation.		
3. SEN 371 and SEN 471 will always be ON if Request to Send is left on at both ends when using 201B1. Carrier On also comes ON when outputting using a 201A3 dataset.		
4. 201A3 = Half-Duplex — 2 wire. 201B1 = Full-Duplex — 4 wire.		

Figure 13-33. Dataset Controllers Instructions (Model 620-65A)

Controller	
General	Provides control, character assembly and disassembly, and interface for one 201A or 201B Data SEt
Construction	Approximately 90 integrated circuits and 10 modular receivers/drivers mounted on a controller quarter board
Code	8-level; conversion accomplished by 520/i software
Input Power	+5 Vdc
Environment	0° to +50°C; 0-90% relative humidity (no condensation)
201A Data Set	
Type	Single Channel, dial-up modem
Speed	2000 bits per second (fixed rate)
Operational Modes	Full and Half Duplex
Data Line	2-wire; two- or four-wire switched line
201B Data Set	
Type	Single Channel, direct-line modem
Speed	2400 bits per second (fixed rate)
Operational Modes	Full and Half Duplex
Data Line	Two- or four-wire, dedicated line
Interconnecting Cable	
Type	12-wire cable with interface connectors for controller and modem
Length	10' standard; available up to 50'

Figure 13-33a. Dataset Controller Specifications (620-65C)

EXTERNAL CONTROL

1000xx	Reset Request to Send
1001xx	Initialize Input
1002xx	Initialize Output
1003xx	Enable Request to Send
1004xx	Reset Error Storage
1005xx	Enable BIC to Input
1006xx	Enable BIC to Output
1007xx	Enable Transparent Mode

DATA TRANSFER

1030xx	Output Memory to M.C. Output Buffer
1031xx	Output A Reg to M.C. Output Buffer
1032xx	Output B Reg to M.C. Output Buffer
1020xx	Input M.C. Input Buffer to Memory
1021xx	Input M.C. Input Buffer to A Reg.
1022xx	Input M.C. Input Buffer to B Reg.
1025xx	Input M.C. Input Buffer to Cleared A Reg.
1026xx	Input M.C. Input Buffer to Cleared B Reg.

SENSE

1010xx	Channel Disabled
1011xx	Input Buffer Ready
1012xx	Output Buffer Ready
1013xx	Carrier On
1014xx	Error (any Error)
1015xx	Parity Error
1016xx	Overflow Error/Underflow Error
1017xx	Control Word

NOTE: xx = 50 for First Modem Controller
 xx = 57 for Eighth Modem Controller

Figure 13-33b. Dataset Controller Instructions (620-65C)

Organization	Contains necessary timing circuitry, registers (in and out) for a complete character, modem control register and EIA drivers and receivers.		
Modes of Operation	Full or half duplex.		
Message Format	Asynchronous.		
*Speed of Operation	Standard speeds: 110, bits per second, 134.89, or 150 BPS. Other speeds (up to 300 BPS) available on request.		
*Character Format	9, 10 or 11 bit character, 1 start, 7 or 8 data and 1 or 2 stop bits.		
Typical Peripheral Devices and Characteristics	Device	Speed in BPS	Character Format
	*Teletypc, Model		
	33 or 35	110	11 Baud
	Teletype, Model 37	150	10 Baud
	IBM 1050 or		
	Mosler Selectriever	134.89	9 Baud
	Usage of other peripheral devices with similar speeds and character formats is possible on request.		
Operations Environment	Operating Temperature = 0 to 50 degrees C.		
	Operating Relative Humidity = 0 to 90%		
*The 620i-66 is wired normally to operate a Model 33 or 35 teletype.			

Figure 13-34. Dataset Controller Specifications (Model 620-66)

Mnemonic	Octal	Functional Description
A. External Control		
EXC 0471	100471	Initialize
EXC 0271	100271	Select Load MCR
B. Transfers		
IME 071	102071	Transfer Read Buffer to Memory
INA 071	102171	Transfer Read Buffer to A Register
INB 071	102271	Transfer Read Buffer to B Register
CIA 071	102571	Transfer Read Buffer to A Register cleared
CIB 071	102671	Transfer Read Buffer to B Register cleared
OME 071	103071	Transfer Memory to Write or MCR Buffer
OAR 071	103171	Transfer A Register to Write or MCR Buffer
OBR 071	103271	Transfer B Register to Write or MCR Buffer Write or MCR
C. Sense		
SEN 0171	101171	Output Buffer Ready
SEN 0271	101271	Input Buffer Ready
SEN 0371	101371	Call Connect
SEN 0471	101471	Call Disconnect
SEN 0571	101571	Carrier On
Notes:		
1. All commands listed are used for 103A2 Dataset operation.		
2. Request to Send and Clear to Send could be substituted for SEN 371 and SEN 471 commands if desired.		
3. 103A = Dialup; 013F = Private Line.		
4. Device address listed is 71; any other device address may be used according to system requirements.		

Figure 13-35. Dataset Controllers Instructions (Model 620-66)

Organization	Contains controller logic, timing circuits, and I/O drivers/receivers to interface with Bell 202 type or equivalent
Modes of Operation	Full or half duplex
Message Format	Asynchronous
Speed of Operation	up to 2000 baud
Character Format	9, 10 or 11 bit character
Priority Assignment	Under program, interrupt or BIC control
Operational Environment	Operating Temperature = 0° to 50°C Operating Relative Humidity = 0 to 90%

Figure 13-36. Dataset Controller Specifications (Model 620-67)

Mnemonic	Octal	Functional Description
A. External Control		
EXC 0	10005X	Initialize
EXC 1	10015X	Connecto BIC to Output
EXC 2	10025X	Set Data Terminal Ready
EXC 3	10035X	Reset Data Terminal Ready
EXC 4	10045X	Set Request to Send
EXC 5	10055X	Reset Request to Send
EXC 6	10065X	Receive Enable
EXC 7	10075X	Receive Disable
EXC B0	10405X	Set Reverse Channel Transmit
EXC B1	10415X	Reset Reverse Channel Transmit
B. Transfers		
OAR	10315X	XFER A Reg. to Controller (transmit)
OBR	10325X	XFER B Reg. to Controller (transmit)
OME	10305X	XFER Memory to Controller (transmit)
INA	10215X	XFER Receive/Controller Data to A Reg
INB	10225X	XFER Receive/Controller Data to B Reg
IME	10205X	XFER Receive/Controller Data to Memory
CIA	10255X	SFER Receive/Controller to Data to cleared A Reg
CIB	10265X	XFER Receive/Controller Data to cleared B Reg
C. Sense		
SEN 0	10105X	Reverse Channel On
SEN 1	10115X	Ring Indicator
SEN 2	10125X	Data Set Ready
SEN 3	10135X	Carrier On
SEN 4	10145X	Clear to Send
SEN 5	10155X	Even Parity
SEN 6	10165X	Transmit Data Ready
SEN 7	10175X	Receive Data Ready

Figure 13-37. Dataset Controller Instructions (Model 620-67)

Automatic Calling Unit Controller

The Model 620-62 Controller is a special-purpose option designed to interface the 620/f computer with the Western Electric Model 801 Automatic Calling Unit (ACU). The ACU and controller permit the 620/f to automatically place a call to any telephone number in the network and then switch the line to a data set for fully automatic transmission of data. Under control of the 620/f and controller, the ACU performs all functions normally performed by an attendant in originating a data call.

The telephone number to be called is stored in the 620/f and presented to the ACU via the controller, bit-parallel, in 4-bit binary digits. The controller contains all control logic and interface required for operation of the ACU under interrupt and sense control.

Organization	Contains a 4-bit output buffer, output interface, and command decoding and interface logic.
Construction	Logic consists of integrated circuits mounted on standard I/O controller board
Device Address	16 (Hexadecimal)
Input	Bit-parallel, buffered by receivers/drivers on I/O tray
Output	Bit-parallel, output interface meets RS-232 specifications
Input Power	+5 Vdc @ 180 mA, -12 Vdc @ 9 mA, +12 Vdc @ 48 mA
Environmental	0° to 50°C; 0-90% relative humidity (no condensation)

Figure 13-38. Automatic Calling Unit Controller Specifications

Mnemonic	Octal	Functional Description
Device Address = 7X		
EXC 0	10007X	Initialize
EXC 1	10017X	Enable CRQ
EXC 2	10027X	Disable CRQ
EXC 3	10037X	Not Used
EXC 4	10047X	Reset DPR — Not normally required for most operating
SEN 1	10117X	Sense PWI modes
SEN 0	10107X	Sense DLO
SEN 2	10127X	Sense ACR
SEN 3	10137X	Sense COS
SEN 4	10147X	Sense PND
OAR	10317X	XFER "A" Reg. to Controller
OBR	10327X	XFER "B" Reg. to Controller
OME	10307X	XFER Memory to Controller

Figure 13-39. Automatic Calling Unit Instructions

Data Communications Controller

The Model 620/f data communications controller system (DCC-1) provides a data communications link between the Varian 620/f computer and remote teletype terminals.

Using telephone lines as the communications media, bidirectional transmission of binary serial data is accomplished by means of dataset modems and 33/35 ASR TTY terminals. The dataset modems must be located within 50 feet (15m) of their associated interface units.

The DCC-1 system offers an economical and efficient means of providing on-line computer services to many remote users. Some typical applications include: on-line program debugging, on-line computation and execution, information storage and retrieval data, inquiry services, computer aid for 'hands-on' use in classroom instruction, and scientific hybrid simulation. In addition, these remote terminals can also be used in an off-line mode for general-purpose utility program routines.

The overall DCC-1 system consists of the power supply (620-95-5), dataset, TTY coupler, 33/35 ASR TTY, a line controller card, and a multiplexer card.

The multiplexer circuit board (DM171) is used to select and control the operation of all the enabled data channels. The line controller circuit board (DM135) provides the logic for four data communication channels.

Organization:	
Line Controller	Contains address gates, clock selection logic, and standard driver/receiver stages
Multiplexer	Contains sense and control function logic, address decode clock and logic circuits, interrupt decode enable logic, and clock timing circuits
Capability:	
Line Controller	Selects address and generates the control logic necessary to synchronize and communicate with dataset modems; each line controller provides four bidirectional data communications channels; and can be expanded to 16 data communications channels
Multiplexer	Provides interface between the computer I/O bus and data communications channels; decodes the address and function data, output function commands, channel ⁴ address codes, and interrupt command signals
Logic Levels	
Positive Logic	
True: +2.5 to +5.5V dc	
False: 0.0 to +0.5V dc	
Size:	
Line Controller	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-circuit card (contains logic for four data channels); system can be expanded to incorporate 12 additional data channels
Multiplexer	One 7.75-by-12-inch (19.7 x 30.3 cm) wired-circuit card
Interconnection	
Interfaces with the computer via a 122-terminal connector; interfaces with external devices via two 44-terminal connectors	
Input Power:	
Line Controller	+5V dc at 300 mA
Multiplexer	+5V dc at 600 mA
Operational Environment	
0 to 50 degrees C, 0 to 90 percent relative humidity without condensation	

Figure 13-40. Data Communication Controller Specification

The controller functions are:

- a. Receive. The controller receives data from the modem and transfers it to the computer.
- b. Transmit. The controller receives data from the computer and outputs it to the modem.

Varian 620/DC Data Communication System

The Varian 620/DC Data Communication System is a computer-controlled, time-multiplexed system that can serve either as a data concentrator, linking a number of low-speed lines to one or more high-speed lines, or as a communications preprocessor, organizing data for direct entry into a large computer.

The design of the Varian 620/DC system represents an optimum balance of hardware and software elements. It is adaptable, without hardware modifications, to virtually every type of communication device now available. The system can also be adapted to various input/output combinations of hardware and software principally through the software interface. As the communication network changes and expands, the Varian 620/DC system can be altered to meet the new requirements by simply revising the computer program.

Both versatility and efficiency are achieved by a functional division of the three major components that comprise the Varian 620/DC system, as illustrated in Figure 1.

The Line Control Modules (LCM) act as an interface between the IC logic levels within the system and the signal levels of the modems that connect to the communication lines. The LCM are the only elements that are specific to the type of terminal or modem connected to the system.

The Varian 620-68 Communications Controller (CC) contains a solid-state IC memory to store separate control information for up to 64 data communication lines. The high-speed IC memory is also used with an 85-bit working (W) register to assemble and disassemble data characters and thus act as a buffer between the serial-bit stream

transmitted by the communication lines and the parallel-bit transfers processed by the computer.

The Varian 620 computer core memory serves as a data reservoir between low-speed and high-speed data communication lines, accepting and transmitting complete character and message blocks at rates compatible with each line.

Since the communications controller and line control modules relieve the computer of many housekeeping functions, such as character assembly and disassembly, synchronization, and parity checking, the computer can be used to perform a variety of data processing functions, such as format changes and recordkeeping.

The unique control information stored in the controller's IC memory for each data communication line makes it possible to process a variety of data signals in a single communication network.

Table 1 summarizes typical loading of a 620/DC system by a variety of data communication devices operating at different data rates.

Versatile Interfacing

The Varian 620/DC can interface with many types of data communication devices. Varian's design philosophy is that selection of the data terminals, modems, and common-carrier lines are the prerogative of the user and the system should be easily adapted in the field.

Rather than dictating the parameters of the communication network, the Varian 620/DC system is sufficiently flexible to adapt itself to user requirements with respect to other network characteristics, such as: data rate, synchronous or asynchro-

data communications

nous transmission, and type of modem (full duplex, half duplex, or simplex).

Data Rate

The data rate is a function of both the data source and the quality of the transmission line.

For example, the standard 33 ASR Teletype unit is limited to approximately 110 bits per second by the mechanical inertia of the keyboard elements. Dial-up telephone lines are limited by the worst-quality line segment that might be encountered between source and destinations; they are rarely used, therefore, at high data rates. By contrast, a private data communication line, with special conditioning at source and destination, can accommodate transmissions at rates up to thousands of bits per second.

The signal transmissions can be synchronous or asynchronous, full-duplex, half duplex, or simplex, with discrete bit rates from 45 to 1200 baud for asynchronous transmission and 45 to 4800 baud for synchronous transmission. Each character in the data stream can consist of five, six, seven, or eight bits, with an odd- or even-parity test generated by the controller circuitry. The data terminals or modems can be integrated-circuit, discrete-circuit, or relay-type devices.

Asynchronous Transmission

Up to four asynchronous data rates are established in the Varian 620/DC system using hardware jumpers. The rate for an individual channel is determined by the control information for that channel. A switch from one of the four rates to another can be accomplished at any time by software, and, if there is a major change in the processing requirements, the four rates established by the hardwired connections

can be changed in less than 5 minutes. For a 64 line system, the maximum asynchronous data rate that can be established for the system is 1200 bits per second, however for systems with fewer lines, the rate can be considerably higher.

Synchronous Transmission

Synchronous signals can range up to 4800 bits per second. Characters are transmitted in a continuous stream, without an interval between characters or any start and stop bits. More information can be carried compared to asynchronous transmission at the same bit rate, but the interfacing equipment is generally more complex.

Each message block starts with a series of unique sync characters. When the receiving equipment acknowledges that it has detected the sync code, a continuous stream of data characters is transmitted. The receiving equipment must divide the stream into eight-bit segments to derive meaningful information.

To receive a synchronous signal, the Varian 620/DC system first loads an appropriate sync character in the IC memory. When the sync character is detected, the system is in sync and proceeds to accept eight-bit segments of incoming data. In the transmission mode, the sync word is repeated until a sync acknowledgement is received from the receiving device. A continuous stream of eight-bit characters is then transmitted.

Type of Modem

Each type of modem or terminal device generates a signal that is characteristic of its logical and electrical design. It is the function of the Varian 620/DC line control module to convert these individual signal levels into standard IC logic levels for processing by the Varian computer and its associated equipment.

Modems	Speed (BPS)	Half Duplex	Full Duplex	Sync/Async
103A/B	110	.06%	.12%	A
103F	300	.18	.36	A
201A	2000	1.25	2.50	S
201B	2400	1.80	3.60	S
202C	1200	.72	1.44	A
X202E	600	.36	.72	A

Figure 13-41. Typical Varian 620/DC System Loading for Common Bell System Modems

Three basic types of LCM are available. With relatively minor variations, these three types can process information to and from most data communication devices now in use.

- Devices designed with RS-232-compatible components are interfaced using the Varian 620-68-1 and Varian 620-68-2 line control modules. These LCM provide interface to four data channels.
- Modems and terminals assembled from discrete or hybrid components require a different type of signal translation, and this is provided by the Varian 620-68-3 line control module. Model 620-68-3A provides eight data channels and 620-68-3B contains circuitry for four channels.
- Relay-type terminals, such as the widely used teletype units, can be interfaced by the Varian 620-68-4 LCM which provides eight channels.

Line Configuration	
Capacity	Four channels (minimum) to 64 channels (maximum) for each 620-68 communications controller
Expansion	Plug-in expandable in four-channel increments to the maximum line capacity. (Relay line control modules are added in eight-channel increments).

Figure 13-42. Varian 620/DC System Specifications (1 of 2)

Line Dynamics

Transmission Type	Half duplex, full-duplex or simplex selected for any line channel under software control.
Transmission Mode	Synchronous speeds from 45 to 4800 baud. Only eight-bit (or multiples of eight) character code format is acceptable. Asynchronous speeds from 45 to 1200 baud. Four user-selectable speeds for any channel under program control (values determined by connector wiring). One start bit and either one or two stop bits. Variable size character codes using five, six, seven, or eight bits per character.

Line Program Features

Parity	Checked on incoming and generated for outgoing data lines under software control. Odd or even parity is program selectable.
Priority	Under software control, any line (or lines) can be designated as high priority.

Line Interface

Design Structure	Line control module is plug-in compatible with Western Electric Modems 103, 201, 202, or commercial equivalents; other modules are compatible with asynchronous two-wire systems, asynchronous relay systems, and synchronous, ground-isolated systems.
Interconnection	Modem cables up to 50 feet.

General Characteristics

Primary Power	115 VAC $\pm$ 10 percent, 60 $\pm$ 2Hz, single phase, up to 25 amperes.
Installation	Mounts in a standard 19-inch RETMA cabinet. No special air conditioning, subflooring, special wiring, or site preparation is required.

Operating Environment

Temperature	0 to 50 degrees C.
Relative Humidity	0 to 90 percent (without condensation).

Figure 13-42. Varian 620/DC System Specifications (2 of 2)

Mnemonic	Octal	Functional Description
EXC 0170	100170	Interface Request (CPU Priority)
EXC 0270	100270	Line Reset
EXC 0370	100370	Read W Register
EXC 0470	100470	System Reset (completely clears CC)
EXC 0570	100570	Clear Interface
EXC 0670	100670	Enable CC Interrupts
EXC 0770	100770	Disable CC Interrupts
EXC 0071	100071	Inhibit Output Interrupts until Input Occurs (Echo)
EXC 0171	100171	Inhibit Output Interrupts
EXC 0271	100271	Output Line Break
EXC 0371	100371	Output all Marks Character (pad character)
EXC 0471	100471	Call Character Disassembly Buffer (output character buffer)
EXC 0571	100571	Call Sequence Controller
EXC 0671	100671	Call Character Assembly Buffer (input character buffer)
EXC 0771	100771	Call Line Identification Field
SEN 0070	101070	Sense Interface Available
SEN 0170	101170	Sense Data Ready
OAR 070	103170	Normal Output from A Register to CC
OBR 070	103270	Normal Output from B Register to CC
OME 070	103070	Normal Output from Memory to CC
OAR 071	103171	Mirror Image Output from A Register to CC
OBR 071	103271	Mirror Image Output from B Register to CC
OME 071	103071	Mirror Image Output from Memory to CC
INA 070	102170	Normal Input from the CC to the A Register
INB 070	102270	Normal Input from the CC to the B Register
IME 070	102070	Normal Input from the CC to the Memory
INA 071	102171	Mirror Image Input from the CC to the A Register
INB 071	102271	Mirror Image Input from the CC to Memory

Figure 13-43. Varian 620/DC Instruction List (1 of 2)

Mnemonic	Octal	Functional Description
CIA 070	102570	Clear & Input Normally to the A Register
CIB 070	102670	Clear & Input Normally to the B Register
CIA 071	102571	Clear & Input Mirror Image to the A Register
CIB 071	102671	Clear & Input Mirror Image to the B Register

Figure 13-42. Varian 620/DC Instruction List (2 of 2)

SECTION 14 — SYSTEM OPERATION

The "operator's console" of the Varian 620/f consists of the front panel of the mainframe enclosure and the teletype that is included in most system configurations. These two elements provide all the controls and displays and printed outputs needed to program, operate, and maintain the computer.

Normal Operation

Very little preparation of the Varian 620/f is required before running an object program. Assuming that the system, including peripherals, is properly installed and connected to a source of ac power, perform the following steps:

1. Turn the power ON with the keyswitch provided on the front panel (Figure 14-1).
2. Load the Bootstrap Loader routine (Figure 14-2), using the switches and controls on the front panel.
3. From the teletype punched-tape reader or the high-speed paper tape reader (depending on the Bootstrap Loader routine used), enter the Binary Load/Dump program (Section 22).
4. Check that the P register is set to the load starting address 0X7600, then load the binary object program, using the same I/O device as in step 3 (see Section 22 for load-and-halt and load-and-execute options).
5. Press the RUN switch on the control panel.

Front Panel Controls

The front-panel controls for the Varian 620/f are illustrated in Figure 14-1 and summarized in the following paragraphs.

Power Switch

The key-operated power switch controls the ac input to the Varian 620/f power supply.

In the PWR OFF position, there is no ac input to the power supply.

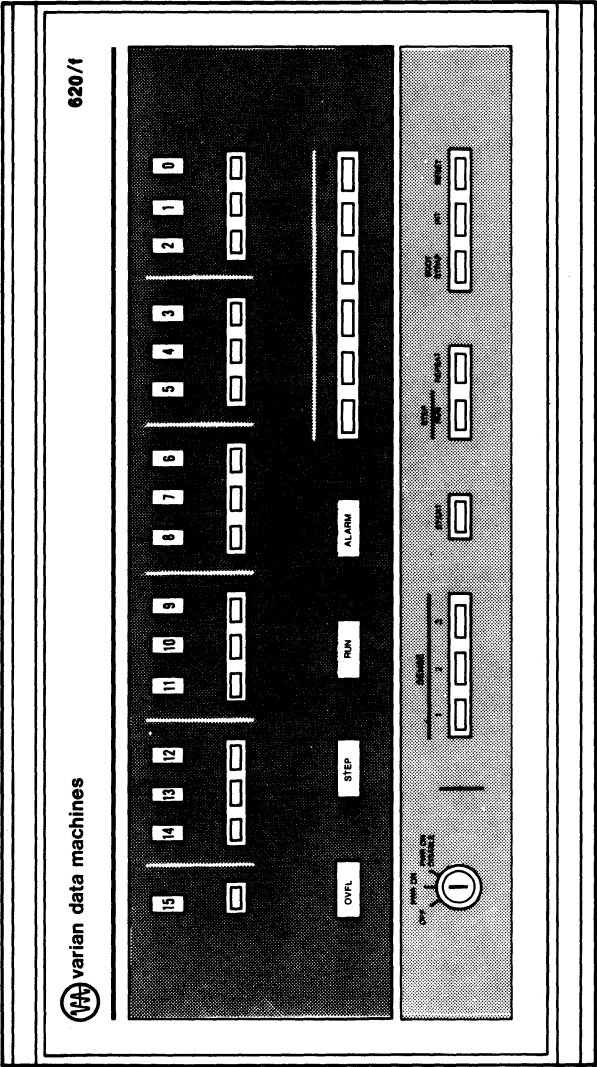
In the PWR ON position, there is ac power to the power supply and the system is fully operational.

In the PWR ON DISABLE position, there is ac power to the power supply and the computer is operational. However, all control console switches are disabled except the power switch itself. Pressing any switch while the power switch is in PWR ON DISABLE has no effect.

The control console and power supply indicator lights are functional when the power switch is in PWR ON DISABLE.

The key can be removed from the power switch in any of the three positions. To turn off the computer, place the power switch in the PWR ON position, lift the STEP/RUN switch, then turn the power switch to PWR OFF.

STEP/run Switch and STEP and RUN Indicators



VT11-703

Figure 14-1. Varian 620/f Front Panel

When the STEP/RUN switch is up, the Varian 620/f is in step mode and the STEP indicator is lit. When the switch is down, the computer is in run mode and the RUN indicator will be lit when the START switch is depressed.

If the computer is in the run mode:

- a. Setting the STEP/RUN switch up to STEP position halts the Varian 620/f after completing the execution of the current instruction and fetches the next sequential instruction and places it in the I register. The RUN indicator goes out and the STEP indicator lights.
- b. Pressing the START switch starts the program at the location specified by the program counter after executing the instruction in the I register.

If the computer is in the step mode:

- a. Setting the STEP/RUN switch down to RUN position puts the computer in run mode. The STEP indicator goes out and the RUN indicator will light when the START switch is pressed.
- b. Pressing the START switch executes the instruction in the I register, and fetches the next instruction from the address specified by the contents of P.

The STEP/RUN switch is in the RUN position for loading the automatic bootstrap and in the STEP position for manual loading of the bootstrap and for manual data entry or register display.

BOOTSTRAP Switch

BOOTSTRAP is a momentary, spring-loaded switch that is functional in Varian 620/f systems containing the optional automatic bootstrap loader. If the option is not included in the Varian 620/f system, the switch is present but is not connected.

The bootstrap program enables the binary load/dump program to be loaded into memory. Before the automatic bootstrap is loaded into memory, the binary load/dump tape should be inserted into the paper tape reader with the first binary frame at the read station.

To load the automatic bootstrap program:

- a. Set the power switch to PWR ON.
- b. Set the STEP/RUN switch to RUN.
- c. Press and release BOOTSTRAP.

When the automatic bootstrap is thus loaded, load the binary load/dump routine provided on paper tape.

If the system does not contain an automatic bootstrap, load the provided bootstrap program manually.

START Switch

START is a momentary, spring-loaded switch. Pressing it when the 620/f is in the run mode starts the program. Pressing the START switch when the computer is in the step mode executes the instruction contained in the I register, and fetches the next instruction from the address specified by the contents of P and places it into the I register.

front-panel controls

REGISTER Switches

Pressing one of the five REGISTER switches selects the designated register (X, B, A, I, or P) for display or entry.

Only one register can be selected at a time. Simultaneously pressing two or more REGISTER switches disables the selection logic to the LOAD switch. The indicators will display the OR of the selected registers.

Register Entry Switches and Display Indicators

The 16 indicators across the top of the Varian 620/f control console display the contents of a selected register. Data are entered into registers on the corresponding register entry switches located under the indicators. The indicators and switches are read from left to right, bits 15 to 0. An illuminated indicator shows that that bit contains a one. For negative data, the sign bit (bit 15) is a one. The indicators and switches are divided into groups of three for ease in reading octal configurations.

Register Display

To display the contents of a register, switch the STEP/RUN switch to STEP and press the REGISTER switch for the desired register.

The display indicators light when they correspond to register bits that contain ones. To remove the display, pull up on the REGISTER switch and the indicators go out.

Data or Instruction Entry

To enter data or instructions in a register:

- a. Display the contents of the register.

- b. Enter ones by pressing down on the register entry switches corresponding to the bits to be set.
- c. Enter zeros in the other bits by pulling up on all other register entry switches. The indicator lights do not change when the register entry switches are manipulated. They still display the contents of the register.
- d. When the desired configuration is entered on the register entry switches, press LOAD. This loads the register with the configuration entered on the switches, and the indicators change to display this new configuration in the register.

To enter data into core memory:

- a. Load into the I register a storage instruction (e.g., STA 054000).
- b. Select the register specified by the storage instruction in step a.
- c. Load the selected register using the data entry switches.
- d. Press START to execute the instruction in the I register. This stores the contents of the specified register in the effective memory location.

LOAD Switch

LOAD is a momentary, spring-loaded switch. When the Varian 620/f is in step mode and a register has been selected, pressing this switch loads the register with the bit configuration entered on the register entry switches.

REPEAT Switch

REPEAT is a toggle switch that is operative in both step and run modes. To repeat an instruction contained in the I register, press REPEAT, and then press START. This instruction is executed again and the program counter advances. However, the contents of the I register remain unchanged.

SENSE Switches

The three SENSE switches are toggle switches permitting program modification by the operator. When the program contains instructions dependent on the setting of these switches, jumps and executions occur when the switch condition is met and do not occur when the switch condition is not met.

To set a SENSE switch, press down. To reset it, pull up. Operations dependent on the position of this switch will be executed if the switch is in the position indicated by the instruction, down being the set position.

Example:

A program can be written so that the operator can obtain a partial total of a column of figures being added by use of the JSS1 (jump if SENSE switch 1 is set) instruction. The program writes individual entries as long as SENSE switch 1 is not set. When the operator wants a partial total, he sets the switch. The program then jumps to an instruction sequence that prints the desired information.

INT (Interrupt) Switch

INT is a momentary, spring-loaded switch used to interrupt the Varian 620/f computer. It is functional only when the Varian 620/f is in the run mode. Pressing INT interrupts to memory location zero.

RESET Switch

RESET is a momentary, spring-loaded switch used as initialization control and for stopping I/O operations. Pressing this switch halts the Varian 620/f and initializes the computer and peripherals. This switch is electrically interlocked with the STEP/RUN switch and is disabled when STEP/RUN is in RUN.

OVFL (Overflow) Indicator

OVFL lights whenever the capacity of the arithmetic unit has been exceeded.

ALARM Indicator

ALARM lights when the local ambient temperature in the mainframe is too high. If the power switch key is accessible, turn the power switch to PWR OFF and call the Varian service engineer.

If the power switch key is not accessible, turn off the power switch located on the back of the power supply, or pull the main plug, and call the Varian customer service engineer.

Cold Start

When a new system is being initialized, or when the contents of memory are unknown, make a cold start:

- a. Turn the power switch key to PWR ON.
- b. Load the bootstrap loader program, either manually (see below) or, if the system contains an automatic bootstrap, automatically.
- c. Load the paper tape binary load/dump program provided. Use a 33/35 ASR Teletype reader or a high-speed reader.

manual operation

To load the bootstrap loader manually:

- a. Load instruction 054000 (Store A Relative to P) in the I register.
- b. Place REPEAT in the down position, and set STEP/RUN to STEP.
- c. Load the starting memory address (007756) of the bootstrap loader into the P register.
- d. Load the bootstrap-loader instruction codes (Figure 14-2) in the A register manually pressing START after loading each instruction. The computer loads the A register contents into the address specified by the P register, which is incremented by one after each instruction is loaded.

To determine that the bootstrap loader has been correctly loaded, perform the following steps.

- a. Initialize the CPU by pressing RESET.
- b. Clear all registers by momentarily pressing each register switch, pressing LOAD each time.
- c. Load instruction 014000 (load A relative to P) in the I register.
- d. Load the starting memory address (007756) in the P register. Keep REPEAT in the down position.

- e. Select the A register and press START. The contents of each memory address is sequentially displayed each time START is pressed.
- f. If an error is found, reload the erroneous instruction codes into memory.

NOTE: The P register error address is always the error address plus one.

Loading Sequential Memory Addresses

To load a sequential group of memory addresses:

- a. Set STEP/RUN to STEP and press REPEAT.
- b. Load the P register with the base address.
- c. Load into the I register a storage instruction (STA, etc.) with 100 in the M field (relative addressing), and zero in the A field.
- d. Select the register specified by the storage instruction in step C.
- e. Load the selected register using the data entry switches.
- f. Press START to execute the instruction in the I register.
- g. Repeat steps e and f until all instructions are loaded. The next cell to be loaded can be observed by displaying the P register.

Displaying Sequential Memory Addresses

To display the contents of a group of sequential memory addresses:

Location	High-Speed Reader*	Teletype*	Symbolic Code
007756	102637	102601	READ, CIB, RDR
007757	004011	004011	, ASLB, NBIT-7
007760	004041	004041	, LRLB, 1
007761	004446	004446	, LLRL, 6
007762	001020	001020	, JBZ, SEL
007763	007772	007772	
007764	055000	055000	, STA, 0, 1
007765	001010	001010	, JAZ, LHLT + 1
007766	007000**	007000**	
007767	005144	005144	, IXR,
007770	005101	005101	ENTR, INCR, 1
007771	100537	102601	SEL, SEL, RDON
007772	101537	101201	, SEN, IBFR, READ
007773	007756	007756	
007774	001000	001000	, JMP, *-2
007775	007772	007772	

*I/O Source for Subsequent Program Inputs

**007600 for Test Executive

manual operation

- a. Place STEP/RUN in STEP, and press REPEAT.
- b. Load the P register with the base address.
- c. Load into the I register a loading instruction (LDA, etc.) with 100 in the M field (relative addressing), and zero in the A field.
- d. Select the register specified by the loading instruction in step c.
- e. Press START once for each memory location to be displayed.
- c. Set the P register to the first address of the program.
- d. Clear the I register.
- e. Press START.
- f. Press START again to execute the instruction and to load the next instruction into the I register.
- g. Repeat step e once for each instruction.

Manual Execution Of A Stored Program

To execute a stored program manually:

- a. Select step mode.
- b. Turn off REPEAT.
- a. Press the REPEAT switch down.
- b. Press START. This advances the P register but inhibits loading the I register. Thus, pressing START again executes the same instruction.

Manual Repetition of Instructions

To repeat an instruction manually:

SECTION 15 – PHYSICAL CONFIGURATION

The basic Varian 620/f computer is contained in a single mainframe chassis that is 10.5 inches high, 21 inches deep, and 19 inches wide. The power supply for the basic computer is contained in a separate chassis, 5.25 inches high, 21 inches deep, and 19 inches wide.

For additional I/O capability and memory expansion, three different types of expansion chassis are available. The expansion chassis are the same size as the mainframe.

Mainframe

The mainframe (Figure 15-1) contains the control panel, a CPU tray, a memory tray, and accommodations for two peripheral controller cards.

The control panel contains all the controls and indicators necessary to operate the computer. The control panel is molded plastic; a printed-circuit card mounted behind the panel holds the lamps and switches. The control panel is hinged to the front of the mainframe. Opening the control panel exposes the CPU and memory trays and allows their removal and installation (Figure 15-2).

The control panel can be located up to 25 feet from the computer mainframe. With a remote control console. The mainframe contains a blank front panel when a remote control panel is used.

The CPU and memory trays each plug into two 100-pin and one 80-pin connectors mounted on a backplane PC board at the rear of the mainframe. The CPU tray also connects to a control panel connector assembled on the end of a flexible cable. As

illustrated in Figure 16-1, the other side of the backplane PC board contains the external rear panel connectors for the mainframe. These connectors allow the mainframe to be interconnected with expansion chassis I and II, PMA controller, TTY unit, and dc power.

Two connector studs on the backplane board distribute ground and +5 volts to the various tray levels.

The CPU tray, as illustrated in Figure 15-3, contains connectors that accommodate 14 circuit cards. These cards contain all of the circuits for the CPU, including the I/O control section and the internal computer options with the exceptions of the PIM and BIC. The CPU tray circuit cards are 14.9 inches long and 3.1 inches wide. Card type 1 can accommodate up to seventy 14-pin dual-in-line sockets; card type 2 can accommodate one 24-pin, four 16-pin, and sixty-four 14-pin DTL sockets. The third type is also used as a PC card for discrete components. The circuit cards contain three card-edge connectors, P1, P2, and P3. Figure 15-4, shows card type 1; types 2 and 3 have the same dimensions.

The memory tray, as illustrated in Figure 15-5, consists of four memory circuit cards approximately 18 inches long and 15.6 inches wide. This configuration applies to both 4K and 8K memory increments. The 8K memory increment requires no more space than a 4K memory increment. When all the memory circuit cards are closed and bolted, the memory tray occupies the same amount of space as the CPU tray.

The two peripheral controller cards are 7.75 inches wide and 12 inches long. These circuit cards (Figure 15-1) are installed side

physical configuration

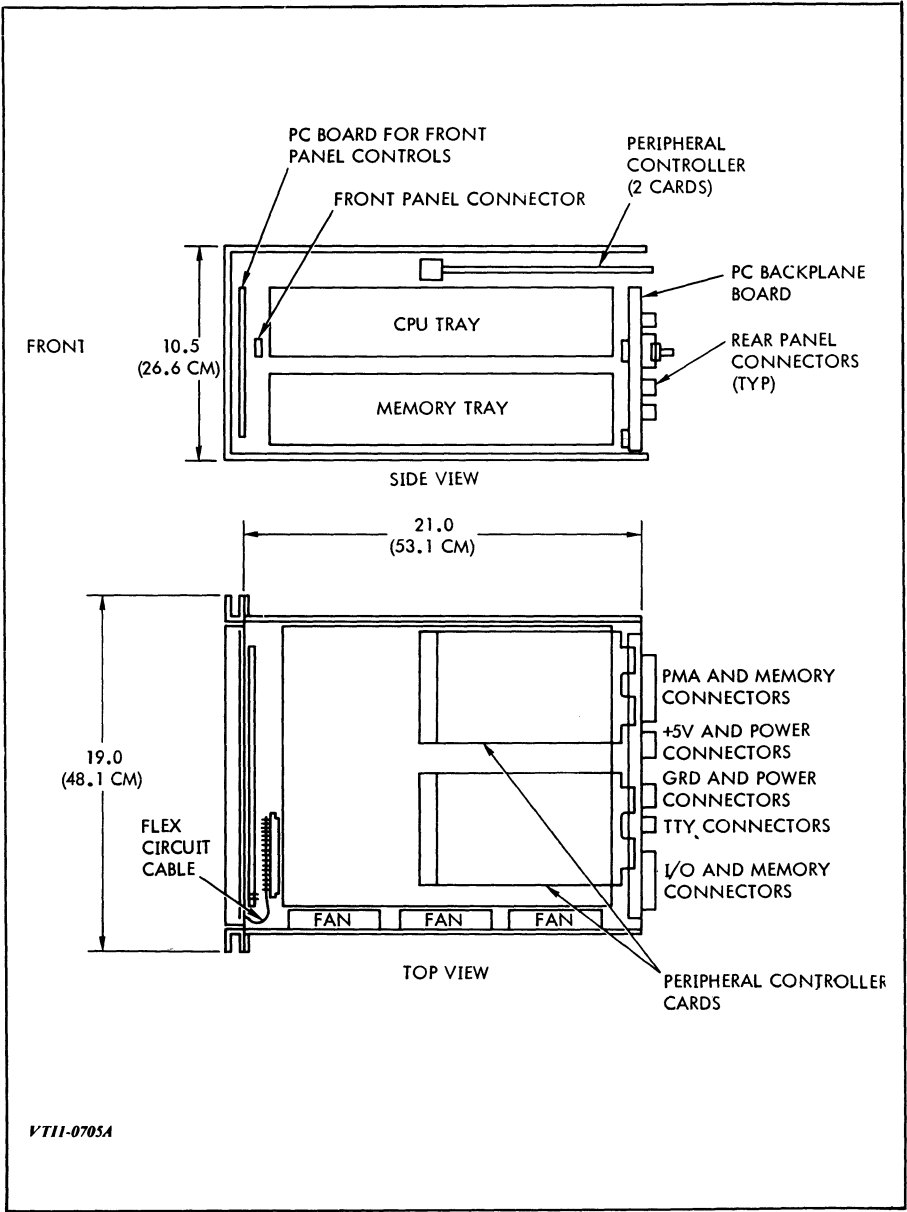
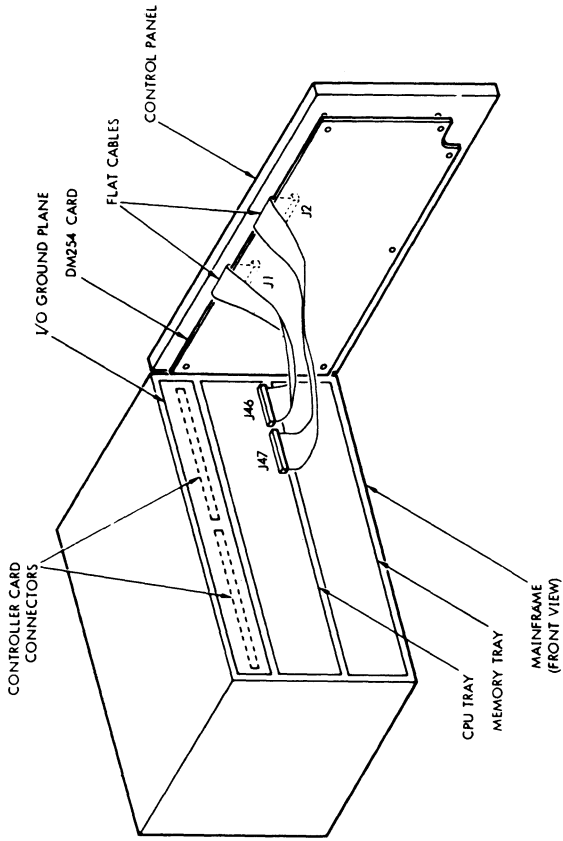


Figure 15-1. Computer Mainframe



V/T11-0967

Figure 15-2. Mainframe With Open Control Panel

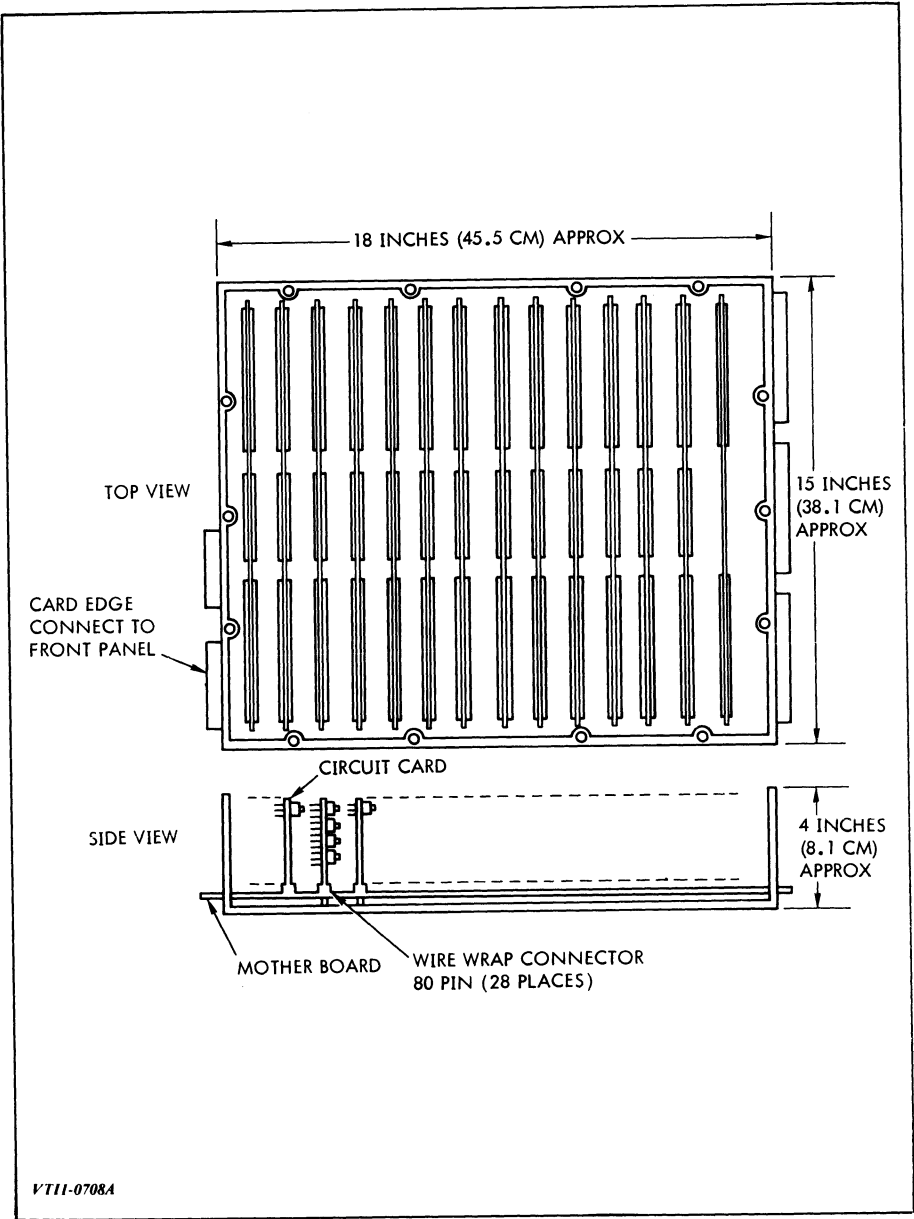


Figure 15-3. CPU Tray Layout

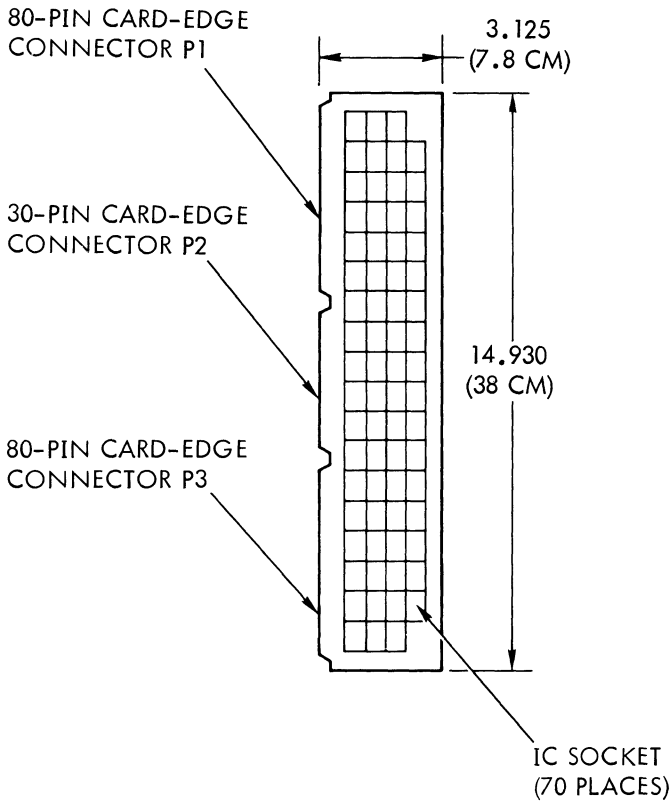


Figure 15-4. CPU Tray Circuit Card

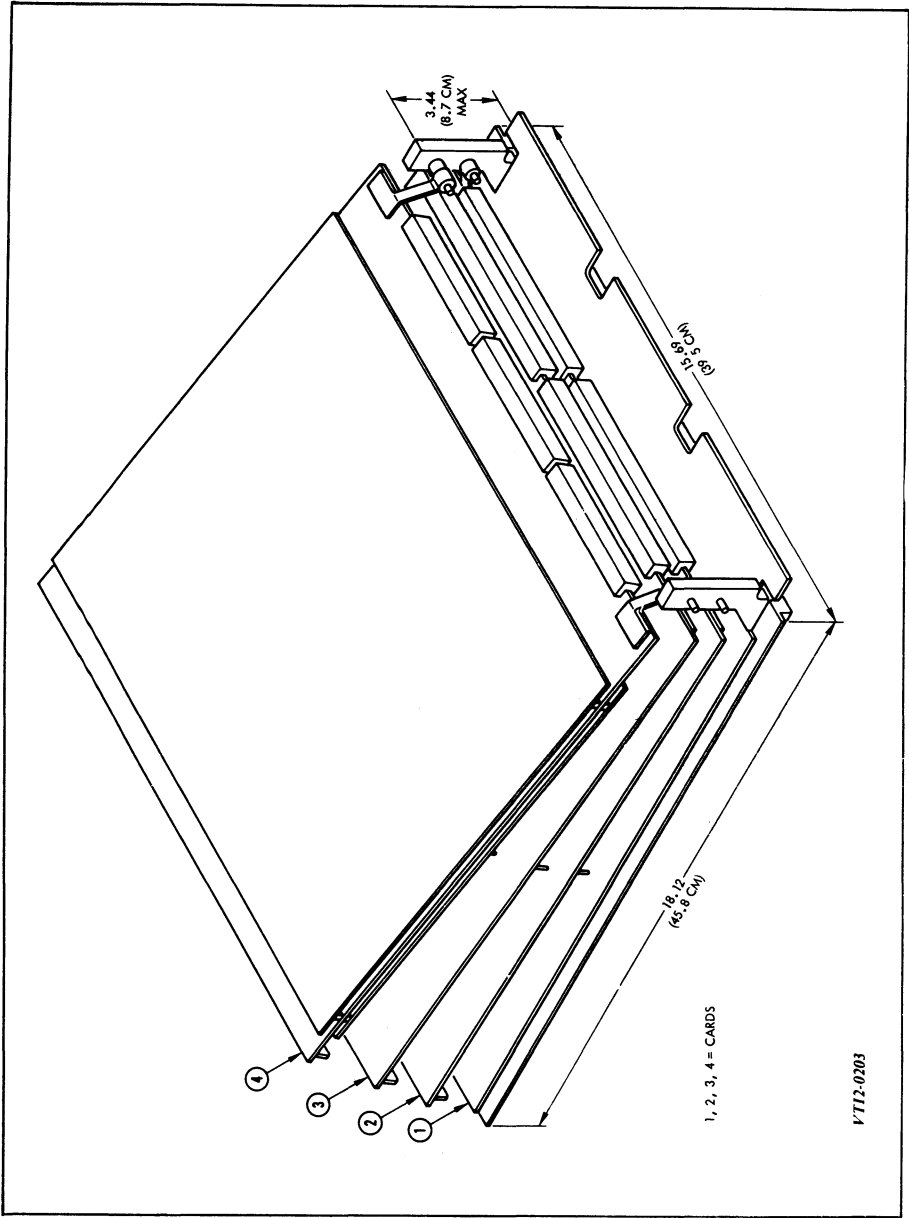


Figure 15-5. Memory Tray

by side through the rear of the mainframe into two 122-pin connectors.

Expansion Chassis

Expansion chassis I, II, and III are available for systems requiring more than 8K of memory or more than two peripheral controller cards. Peripheral controllers are accessed through the rear of the expansion chassis.

Expansion chassis I (Figures 15-6 and 15-7) can contain a single memory tray and up to six peripheral controller cards. Memory tray removal and installation is accomplished through the front of the expansion chassis when the hinged front panel is opened. Peripheral controller cards are installed through the rear of the expansion chassis. A computer system with expansion chassis I can contain a 16K memory and up to eight peripheral controllers.

Expansion chassis II (Figures 15-8 and 15-9) can accommodate the same number of modules as the mainframe. In expansion chassis II, however, an additional memory tray is installed in place of the CPU tray. The front panel of the expansion chassis is hinged as in the mainframe. Opening the front panel exposes the two memory trays and allows for removal and installation. A computer system with a mainframe and an expansion chassis II can contain a 24K memory and four peripheral controllers.

Expansion chassis III permits the addition

of up to 24 peripheral controllers to systems requiring large numbers of peripheral devices.

A typical system installation, illustrating the expansion capability, is shown in Figure 15-10. A mainframe power supply is required for the computer and expansion chassis I and II. A second power supply is required for additional expansion chassis.

Power Supplies

The computer power supplies connect to a standard 115-volt, single-phase, ac power source. Also available, for European installations, are power supplies that operate on 220V ac. Power regulation is not required under normal commercial power conditions. The mainframe power supply draws approximately 15 amperes, and the expansion power supply, approximately 4 amperes.

The dimensions of the mainframe power supply are shown in Figure 15-11.

Environmental Requirements

Ambient temperature at the installation site can vary between 0 and 50 degrees C with no adverse effects on computer operations. To extend the life expectancy of the computer, however, it is recommended that ambient temperature be maintained between 13 and 29 degrees C. The humidity can be up to 90 percent as long as there is no condensation.

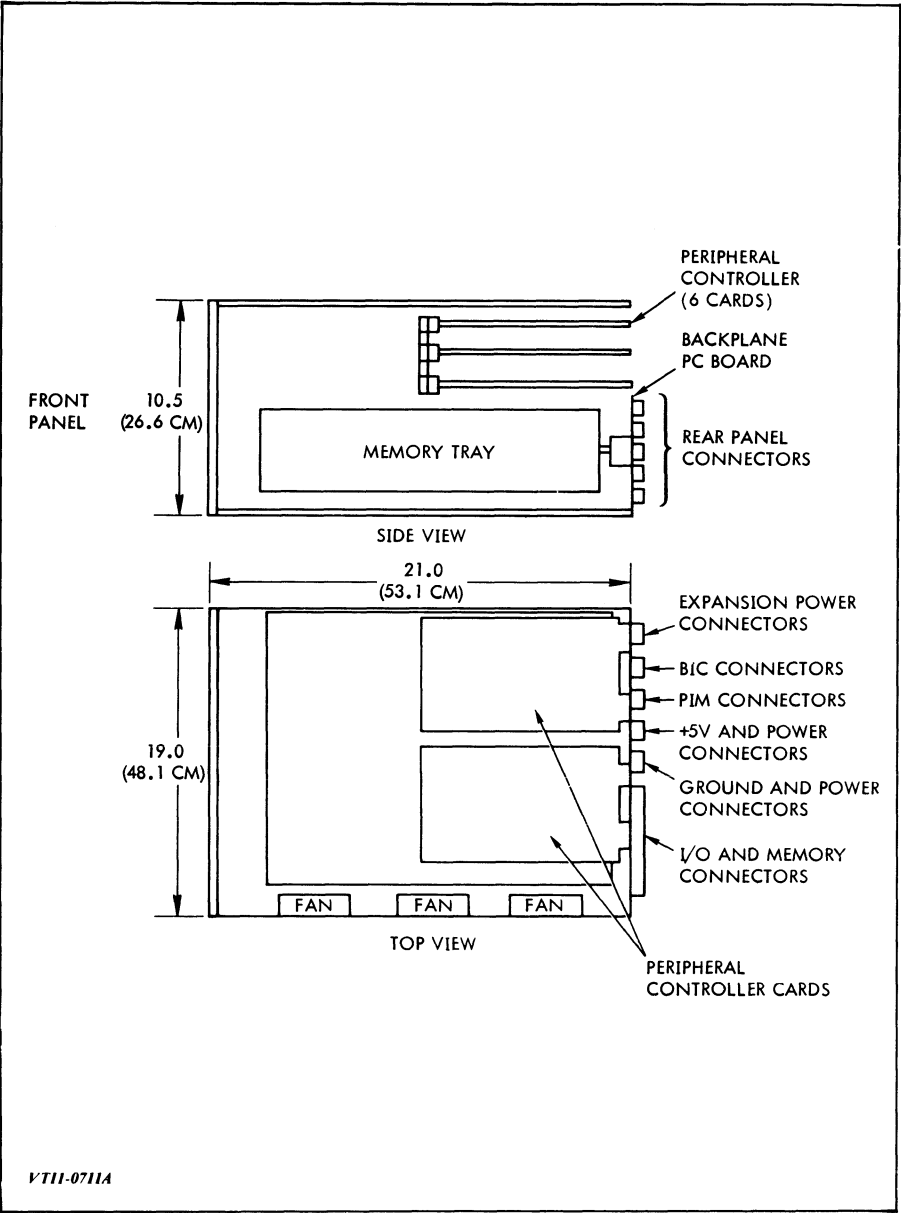
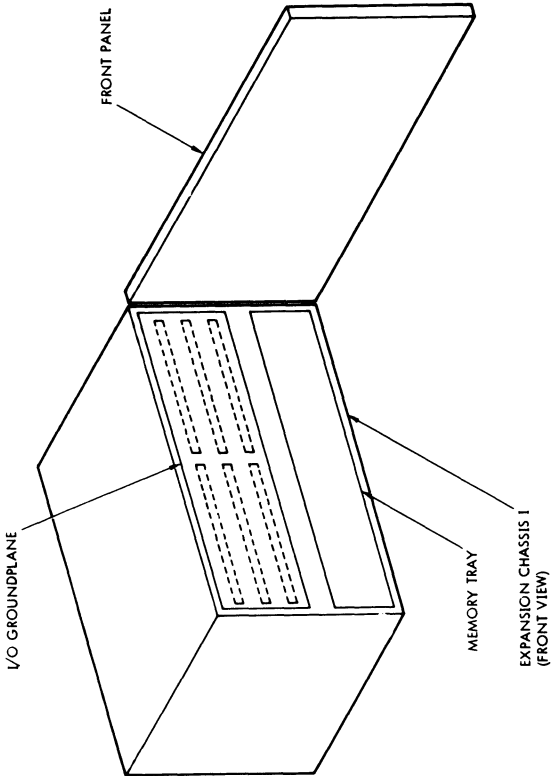


Figure 15-6. Expansion Chassis I



VT11-0975

Figure 15-7. Expansion Chassis I With Front Panel Open

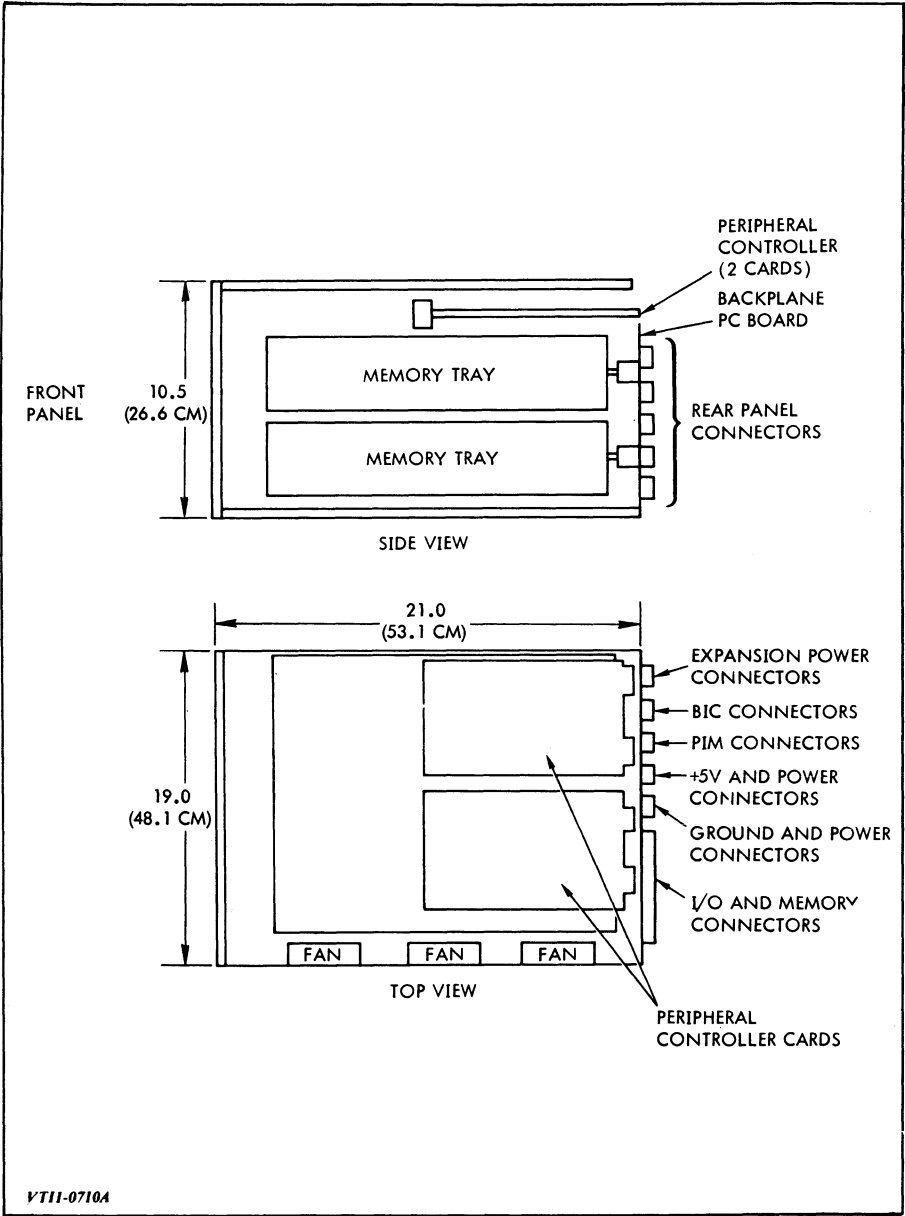
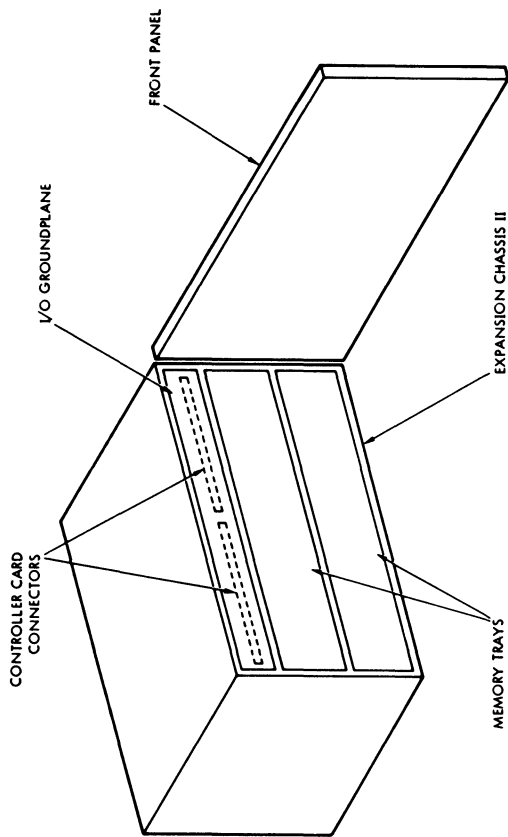


Figure 15-8. Expansion Chassis II



VTII-0979

Figure 15-9. Expansion Chassis II With Front Panel Open

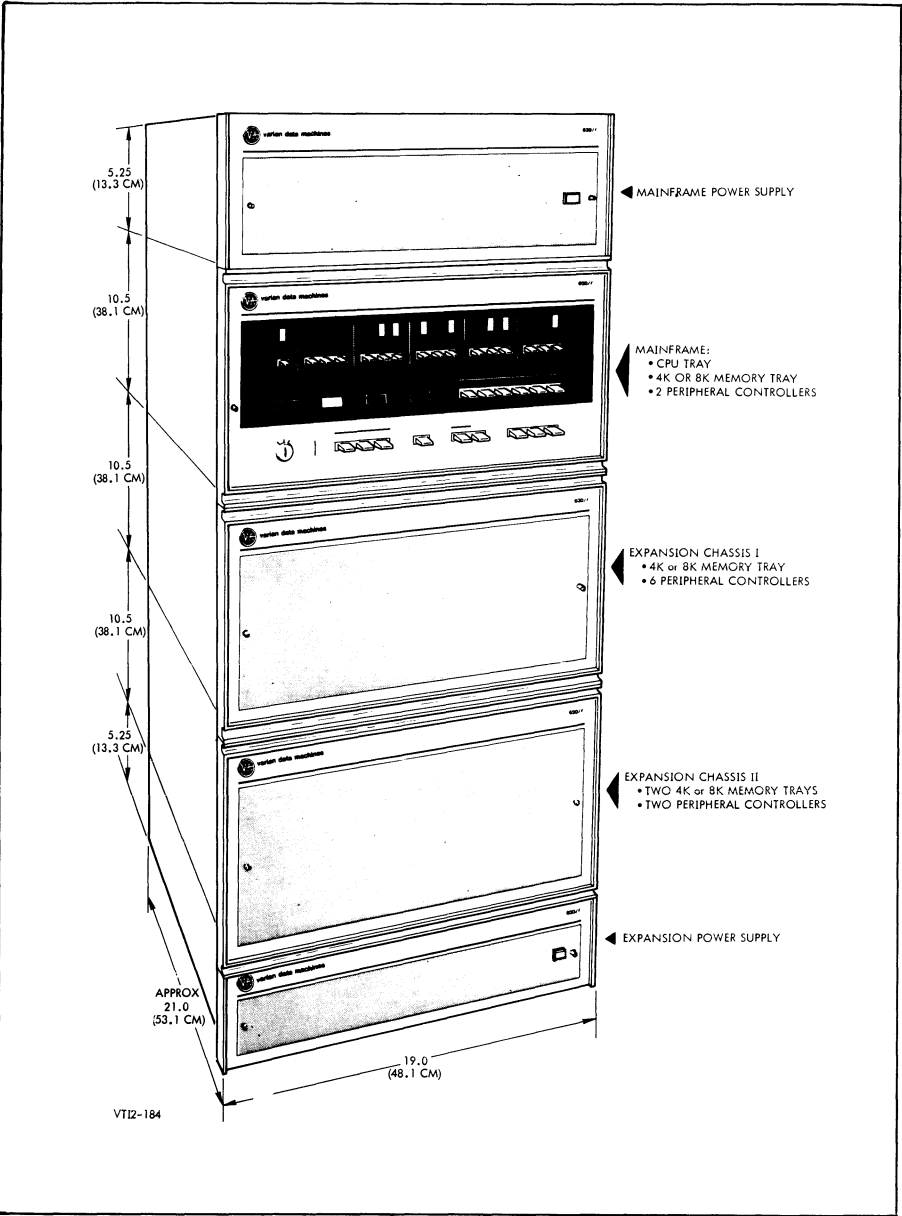
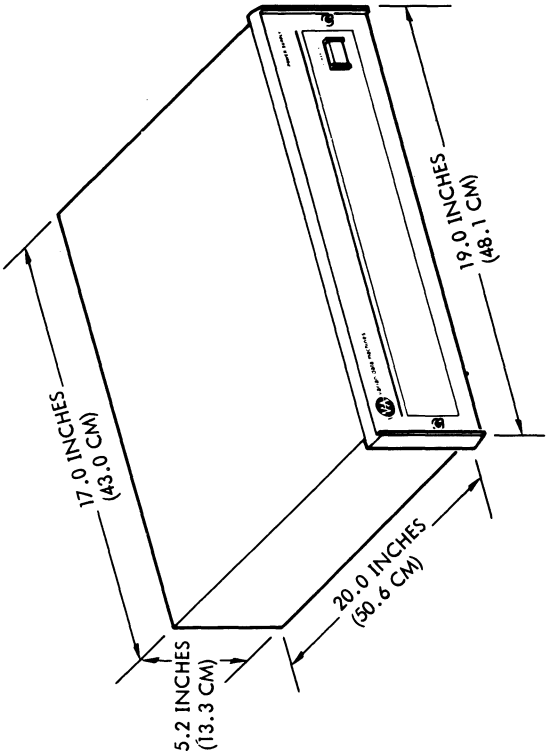


Figure 15-10. Typical System Installation



VTII-0993

Figure 15-11. Varian 620/f Mainframe Power Supply

SECTION 16 – SYSTEM INTERCONNECTIONS

A Varian 620/f system is interconnected by cables that mate with connectors at the rear of the mainframe and expansion frames.

Figures 16-1, 16-2 and 16-3 identify external connectors on the mainframe and expansion chassis backplanes. A typical system interconnection is illustrated in Figure 16-4.

Memory Expansion Interconnection

Two memory expansion cables, left-hand (Part Number 53P0471) and right-hand (Part Number 53P0464), are required for each expansion chassis containing memory. The left-hand memory cable connects J11 on the mainframe to either J18 on expansion chassis II (Model 620/f-011) or J13 on expansion chassis I (Model 620/f-010). The right-hand memory cable connects J12 on the mainframe to either J19 on expansion chassis II (Model 620/f-011) or J16 on expansion chassis I (Model 620/f-010). Memory expansion cables also interconnect two expansion chassis (memory-out connector to memory-in connector).

The maximum length of memory expansion cable in a system is 3 feet. In a system consisting of a mainframe and two expansion chassis containing memory, the length of the memory expansion cables between chassis is 1.5 feet. The connector at each end of the memory expansion cables consists of a 100-pin printed-circuit (PC) card.

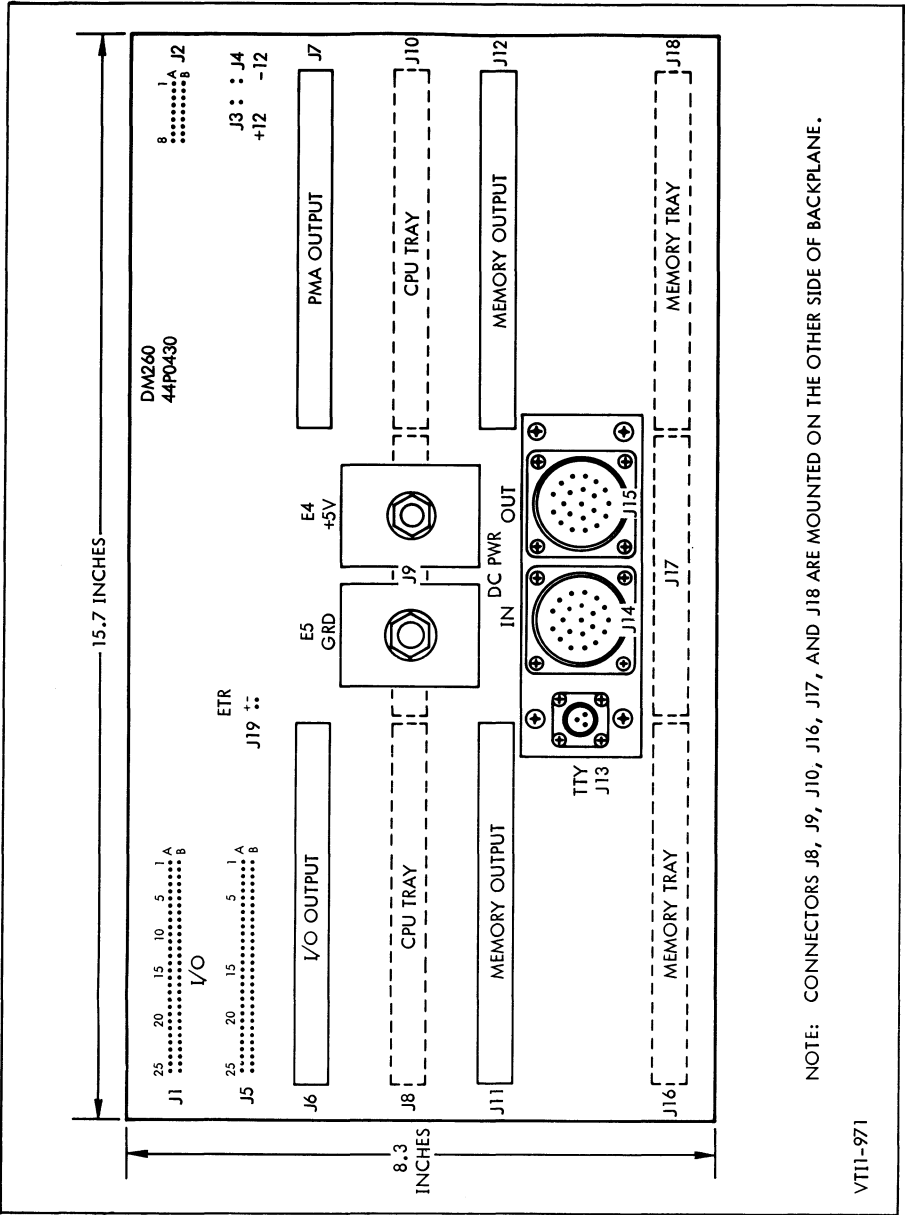
Memory termination shoes are installed on the memory-out connectors (J20 and J21 or J17 and J18) of the last expansion chassis of the system. The left-hand memory termination shoe is Part Number 44P0485 and the right-hand, 44P0486.

I/O Expansion Interconnection

One I/O expansion cable (Part Number 53P0494) is required for interconnecting expansion chassis containing I/O controller cards. The I/O expansion cable connects I/O connector J6 on the mainframe to I/O-in connector J5 on expansion chassis II or expansion chassis I. The same type of I/O expansion cable also connects any two expansion chassis (I/O-out connector to I/O-in connector). Total length of I/O expansion cables can be up to 20 feet. Each I/O cable contains a 100-pin PC card connector at each end. An I/O termination shoe goes on the I/O-out connector of the last expansion chassis of the system.

At the mainframe, the I/O lines routed through connector J6 connect, via etched conductors, to connectors J1 and J5. J1 and J5 (50-pin Berg connectors) connect to two flat I/O cables. As illustrated in Figure 16-5, the flat I/O cables routed in the mainframe to the front of the I/O ground plane are fanned out and connected with Berg pins to E bus pins. The other side of the I/O ground plane contains two circuit-card connectors that accommodate I/O controller cards.

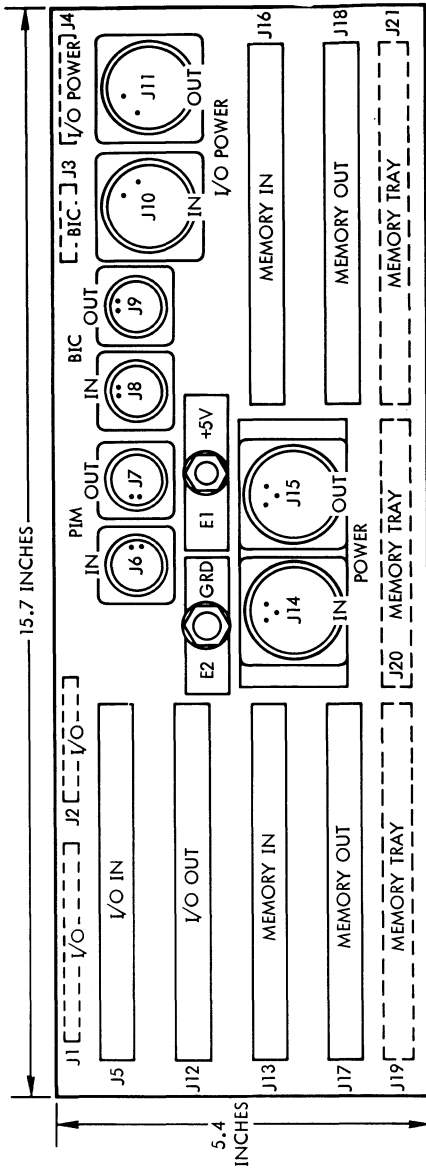
At expansion chassis II, the I/O lines routed through connector J5 connect, via etched conductors, to connectors J1 and J2. J1 and J2 (50-pin Berg connectors) connect to two flat I/O cables. As illustrated in Figure 16-6, the flat I/O cables routed in expansion chassis II to the front of the I/O ground plane are fanned out and connected with Berg terminals to E bus pins. The other side of the I/O ground plane contains two circuit-card connectors that accommodate I/O controller cards.



NOTE: CONNECTORS J8, J9, J10, J16, J17, AND J18 ARE MOUNTED ON THE OTHER SIDE OF BACKPLANE.

VT11-971

Figure 16-1. Mainframe Backplane



NOTE: CONNECTORS J1, J2, J3, J4, J19, J20, AND J21 ARE MOUNTED ON THE OTHER SIDE OF BACKPLANE.

VT11-977

Figure 16-2. Expansion Chassis I Backplane

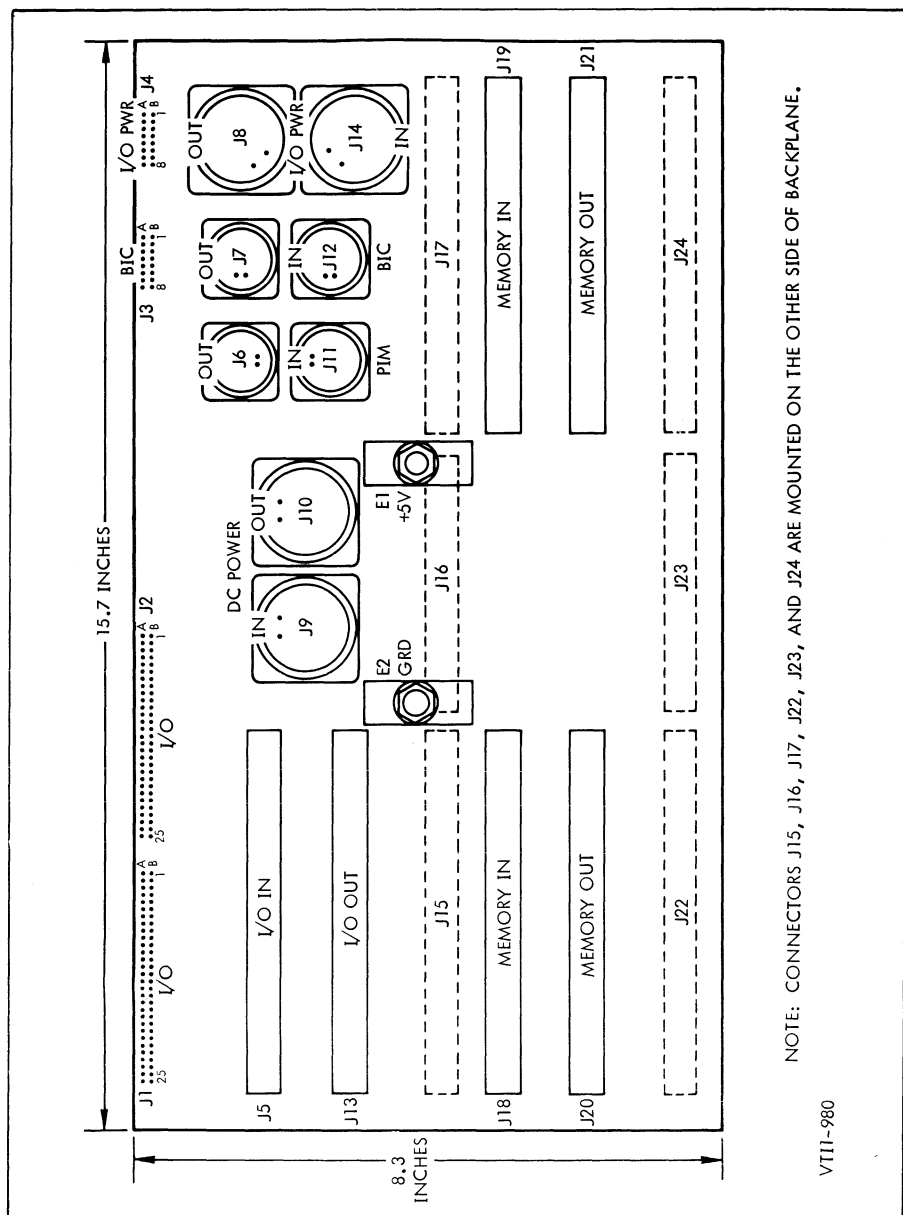


Figure 16-3. Expansion Chassis II Backplane

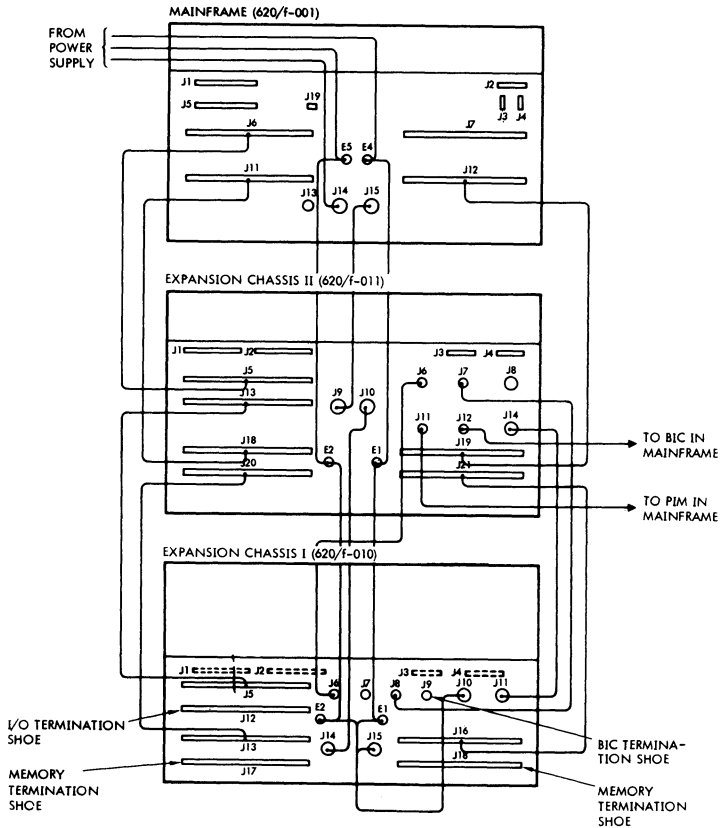


Figure 16-4. Typical System Interconnection

system interconnections

At expansion chassis I, the I/O lines routed through connector J5 connect, via etched conductors, to connectors J1 and J2 (mounted on the other side of the DM276 backplane). J1 and J2 (50-pin Berg connectors) connect inside the expansion chassis to two flat I/O cables. As illustrated in Figure 16-7, the flat I/O cables routed in expansion chassis I to the front of the I/O ground plane are fanned out and connected with Berg terminals to E bus pins. The other side of the I/O ground plane contains six circuit-card connectors that accommodate I/O controller cards.

Power Supply Interconnection

Power for the CPU tray, 32K memory, and ten peripheral controller cards can be supplied by one mainframe power supply.

The connector panel for the power supply is shown in Figure 16-8. As illustrated in Figure 16-4, three cables connect the power supply and the mainframe. One cable routes

all voltages, except +5 volts, to the mainframe. This cable is 2.5 feet long and mates with a 21-pin Cannon connector (J14) on the mainframe. Two 6-gage battery cables, 2 feet long each, supply +5 volts and ground to the mainframe. The +5-volt and ground cables connect to the mainframe at terminal studs E4 and E5, respectively.

To route power to expansion memories (Figure 16-9), expansion power cables connect from mainframe power connectors E4, E5, and J15 to expansion chassis connectors E1, E2, and J9 (or J14), respectively. The expansion power cable (Part Number 53P0466), which provides all voltages except +5 volts, is 2.5 feet long with a 21-pin Cannon connector at each end. This cable connects the power-out connector (J15) on the mainframe to power-in connectors (J9 or J14) on expansion chassis.

The expansion power cable can also connect the power-out connector (J10 or J15) of an

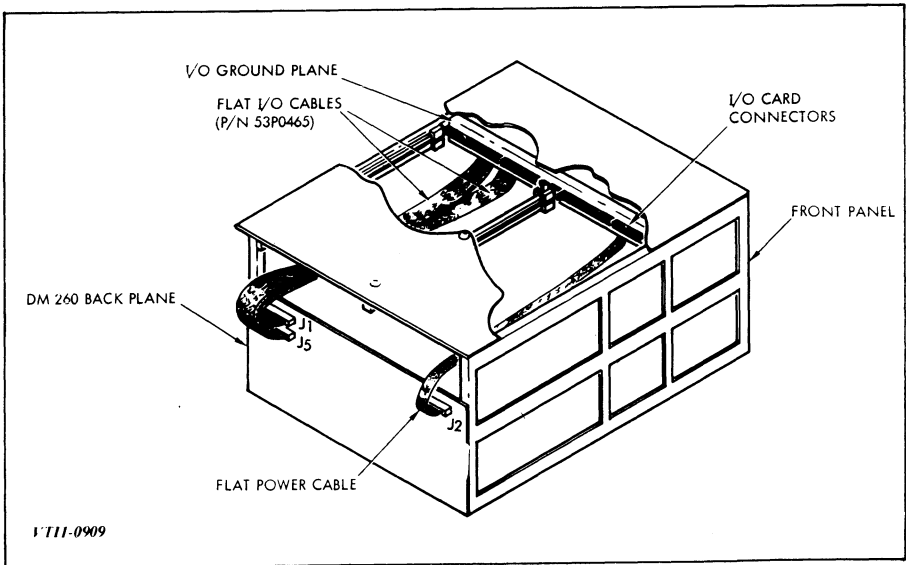


Figure 16-5. Mainframe Internal Cables

expansion chassis to the power-in connector (J9 or J15) of a second expansion chassis. The expansion power cables for +5 volts and ground are 6-gage battery cables, 2 feet long, with terminal lugs at each end. These cables connect mainframe terminals E4 and E5 to expansion chassis terminals E1 and E2, respectively; they can also route +5 volts and ground between two expansion chassis.

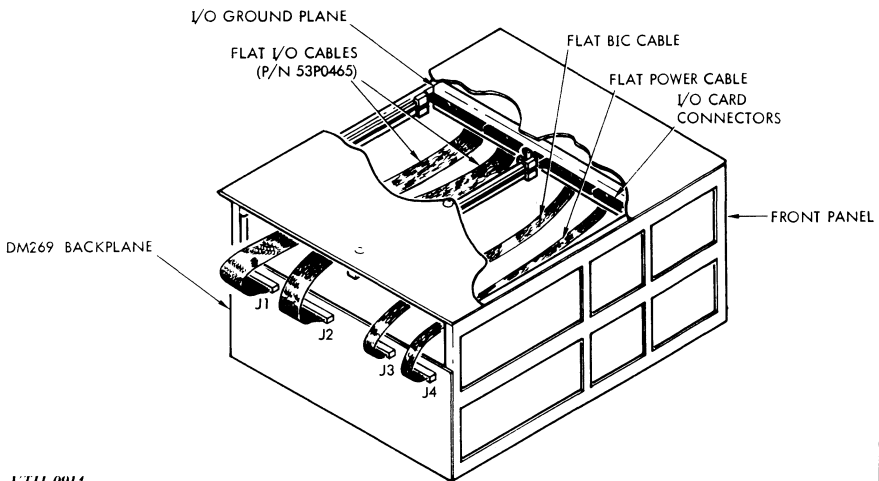
Power for peripheral controller cards in the mainframe is routed to the I/O ground plane in a flat power cable that connects to a 16-pin Berg connector J2 (Figure 16-5). The voltages are routed to connector J2 via etched conductors from J15, E4, and E5.

Power for peripheral controller cards in the expansion chassis is routed through one cable that connects from the three power connectors (E1, E2, and J15 or J10) to I/O-power-in connector J10 on expansion chassis I or J14 on expansion chassis II (Figures 16-2, 3, and 4). The dc power lines routed through I/O-power-in connectors J14

and J10 are connected via etched conductors to power-out connectors J8 and J11, respectively. To supply dc power to I/O controller cards in the second expansion chassis, a 2.5-foot expansion power cable (Part Number 53P0479) connects the I/O power-out connector of one expansion chassis to the power-in connector of the second expansion chassis (Figure 16-4).

All power-in and power-out connectors are 21-pin Cannon. In each expansion chassis backplane, the dc power lines are routed to Berg connector J4 via etched conductors (in expansion chassis I, J4 is mounted on the other side of the backplane). As illustrated in Figures 16-5 and 16-6, J4 connects to a flat power cable that routes dc power to the expansion chassis I/O ground plane.

More than ten peripheral controller cards in a system require an additional expansion chassis I and an expansion power supply. A



VTH-0914

Figure 16-6. Expansion Chassis II Internal Cables

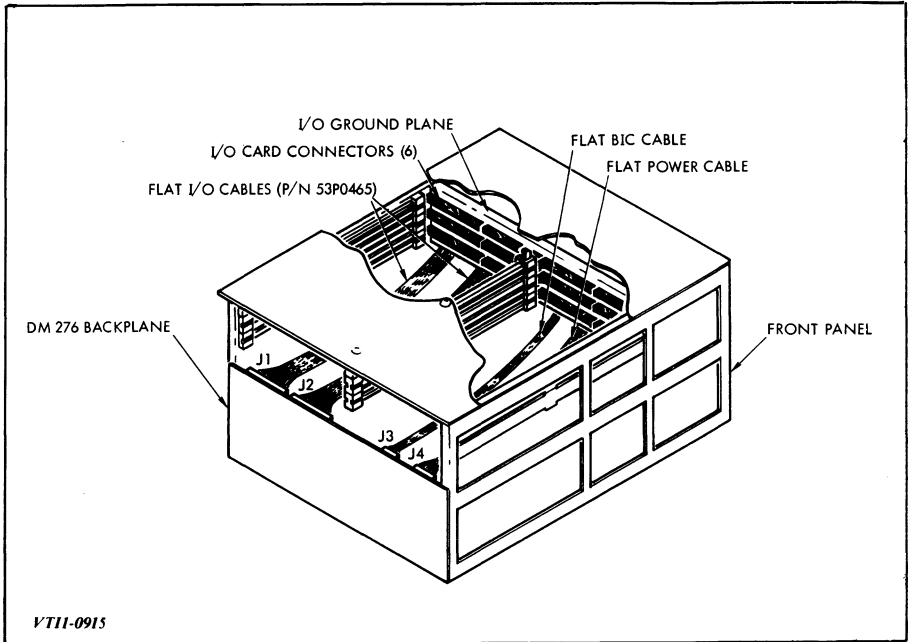


Figure 16-7. Expansion Chassis I Internal Cables

typical computer system containing more than ten peripheral controller cards is illustrated in Figure 16-10. In this example, the second expansion chassis I is interconnected with an I/O expansion cable (connects to I/O-in connector J5) and a cable from an expansion power supply (connects to I/O power-in connector J10).

PJM Interconnection

The optional PJM has eight interrupt lines (IL00 through IL07) for connection with up to eight peripheral controllers. PJM interconnection consists of connecting these interrupt lines to selected peripheral controller cards thus providing them with interrupt capabilities on a priority basis. For peripheral controllers that are located in the same computer chassis as the PJM, the

interrupt lines are connected on the I/O ground plane. For peripheral controllers located in a different chassis, the interrupt line connections are routed through external interrupt cables.

The PJM card is normally installed in one of the I/O card slots of the mainframe. The interrupt lines are routed to the peripheral controller cards in expansion chassis as follows (Figure 16-4): A PJM 1 cable connects to one of 44-pin card-edge connectors J1 or J2 (parallel-wired) of the PJM card. The other end of the cable connects to PJM-in connector J11 on expansion chassis II or J6 on the expansion chassis I. Interrupt lines are routed between two expansion chassis via a PJM 2 cable with an 18-pin Cannon connector at each end. This cable connects a PJM-out connector (J6 or J7) of

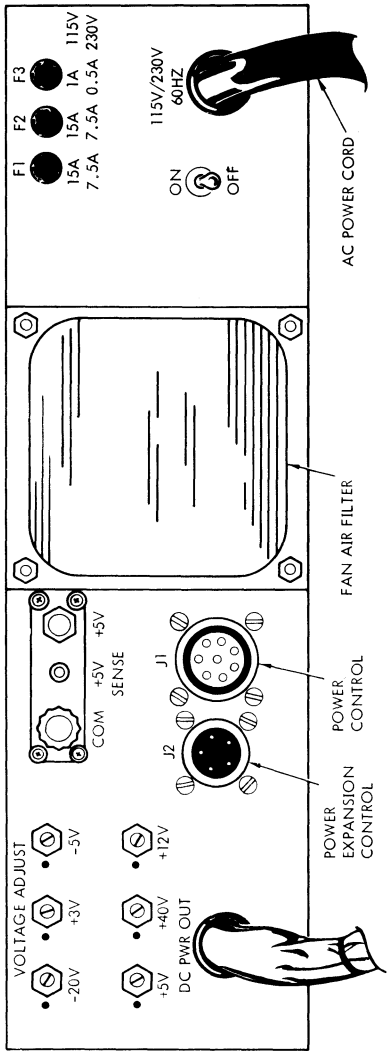


Figure 16-8. Connector Panel, Mainframe Power Supply

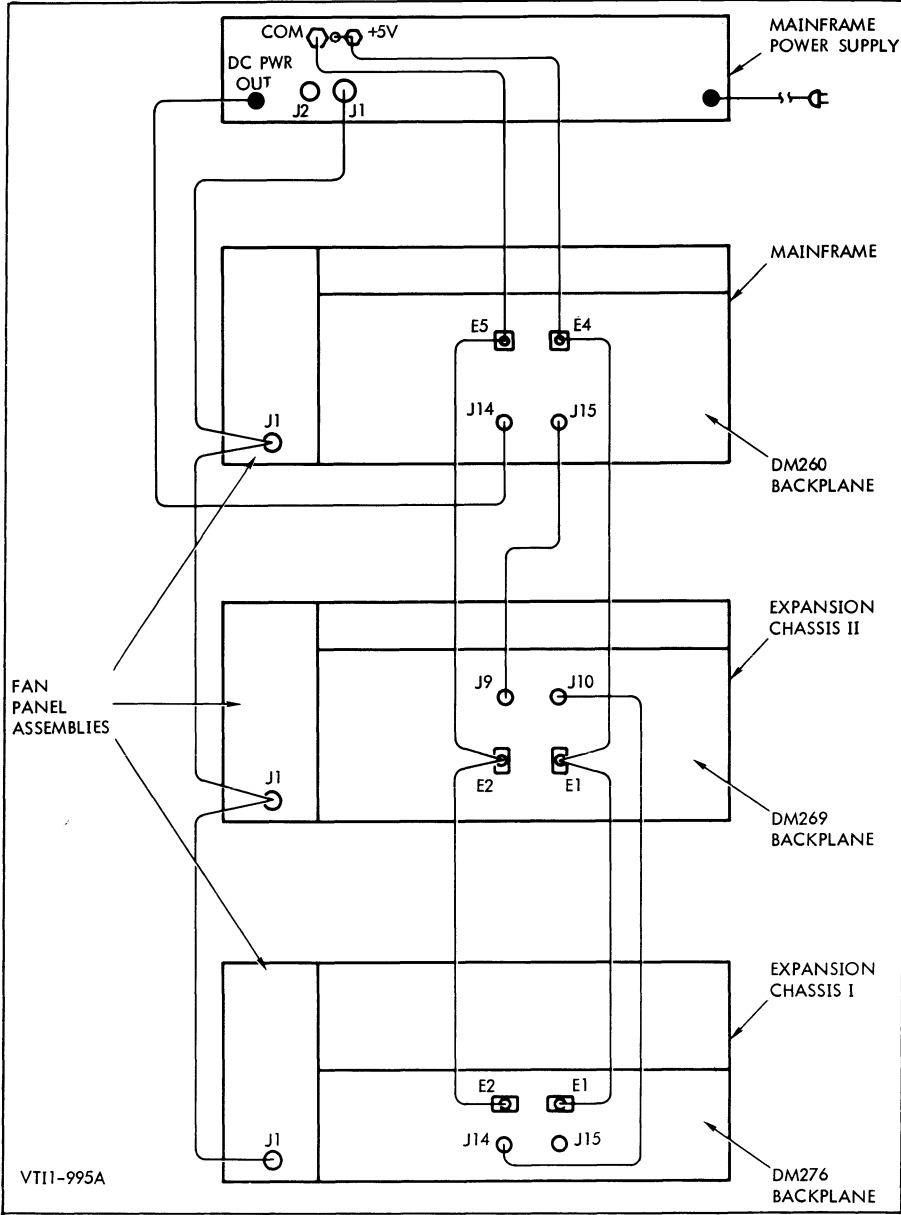
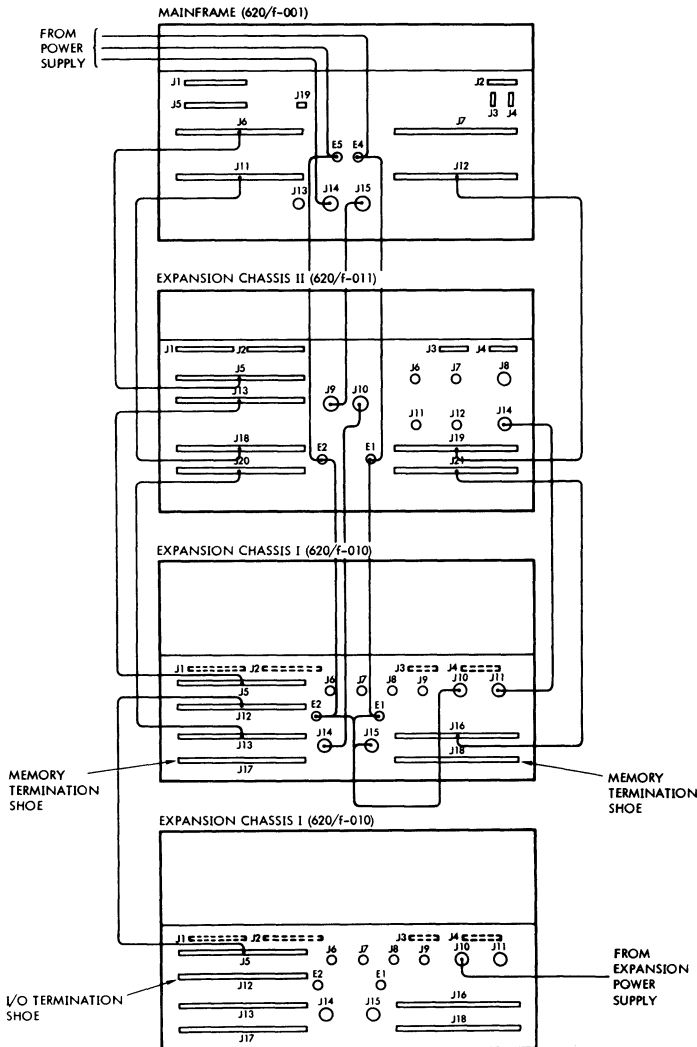


Figure 16-9. Power Connections For 32K System



VT12-0206

Figure 16-10. Typical Expansion Power Supply Interconnection

system interconnections

one expansion chassis to a PIM-in connector (J11 or J6) of a second expansion chassis. All expansion chassis PIM connectors are 18-pin Cannon connectors. Total length of interrupt cables can be 20 feet.

In the mainframe and expansion chassis, the interrupt lines are routed from the backplane to the I/O ground planes via the flat I/O cable. In the mainframe, the interrupt lines are in the flat cable that connects to Berg connector J1 (Figure 16-5); in the expansion chassis, they are in the flat cable that connects to Berg connector J2 (Figures 16-6 and 16-7).

BIC Interconnection

Control signal routing for the optional BIC is divided into two parts. One part consists of BIC/computer interface signals; the other, BIC/peripheral interface signals (B cable signals). For a BIC in the mainframe, the BIC/computer interface signals are routed via the I/O ground plane in the mainframe. For a BIC in an expansion chassis, the BIC/computer interface signals are routed to the mainframe via the I/O expansion cable. For a BIC connected to a peripheral controller (or controllers) in the same computer chassis, the B cable signals are routed via the I/O ground plane; if the BIC is in another chassis, the B cable signals are routed via an external B cable.

Up to four BICs can be installed in a system. When two or more BICs are located in the same chassis, the B cable lines are only connected to the peripheral controller (or controllers) with which each BIC is communicating. B cable lines are not installed between two BICs.

A typical BIC interconnection is illustrated in Figure 16-4. In this example, the BIC card (which is installed in one of the I/O card slots of the mainframe) is connected to

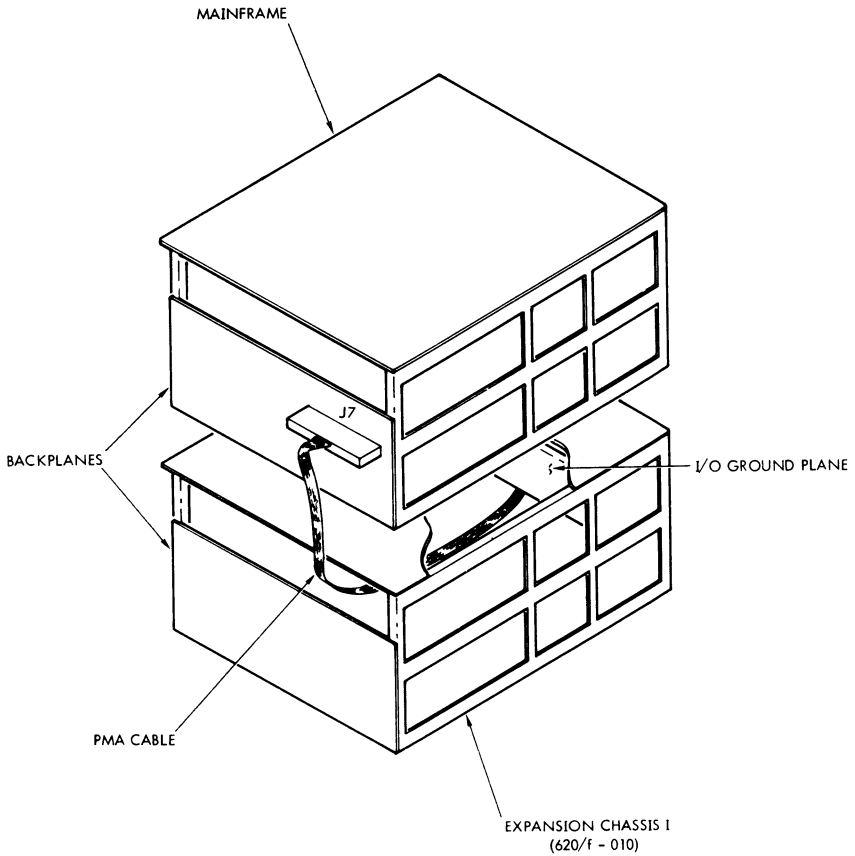
peripheral controllers located in the two expansion chassis. A BIC 1 cable connects to one of 44-pin card-edge connectors J1 and J2 (parallel-wired) of the BIC card. The other end of the BIC 1 cable connects to BIC-in connector J12 on expansion chassis II. The B cable signals are routed to peripheral controllers in expansion chassis I via a BIC 2 cable with an 18-pin Cannon connector at each end. This BIC 2 cable connects BIC-out connector J7 on expansion chassis II to BIC-in connector J8 on expansion chassis I. BIC-out connector J9 on expansion chassis I is terminated with a BIC termination shoe because it is the last BIC connector in the illustrated system.

Except for Berg connector J3, the expansion chassis BIC connectors are 18-pin Cannon connectors. The external BIC cables contain eight twisted pairs and are as short as the system configuration permits. The B cable signals are routed to the I/O ground plane of the expansion chassis in a flat cable that connects Berg connector J3 (Figures 16-5 and 16-6). The signals are routed to the J3 connectors via etched conductors from the BIC-out connectors.

PMA Interconnection

The optional PMA allows up to eight peripheral devices direct access to the Varian 620/f memory. The controllers for the peripheral devices can only be installed in the I/O card slots of expansion chassis I (Model 620/f-010). PMA interconnection consists of routing signals from the PMA circuit in the mainframe to the I/O ground plane in expansion chassis I.

As illustrated in Figure 16-11, a PMA cable (Part Number 53P0513) connects 100-pin card-edge connector J7 on the mainframe to circuit card connectors on the I/O ground plane of expansion chassis I. The PMA lines on the I/O ground plane must also be



VT11-0916

Figure 16-11. PMA Interconnection

system interconnections

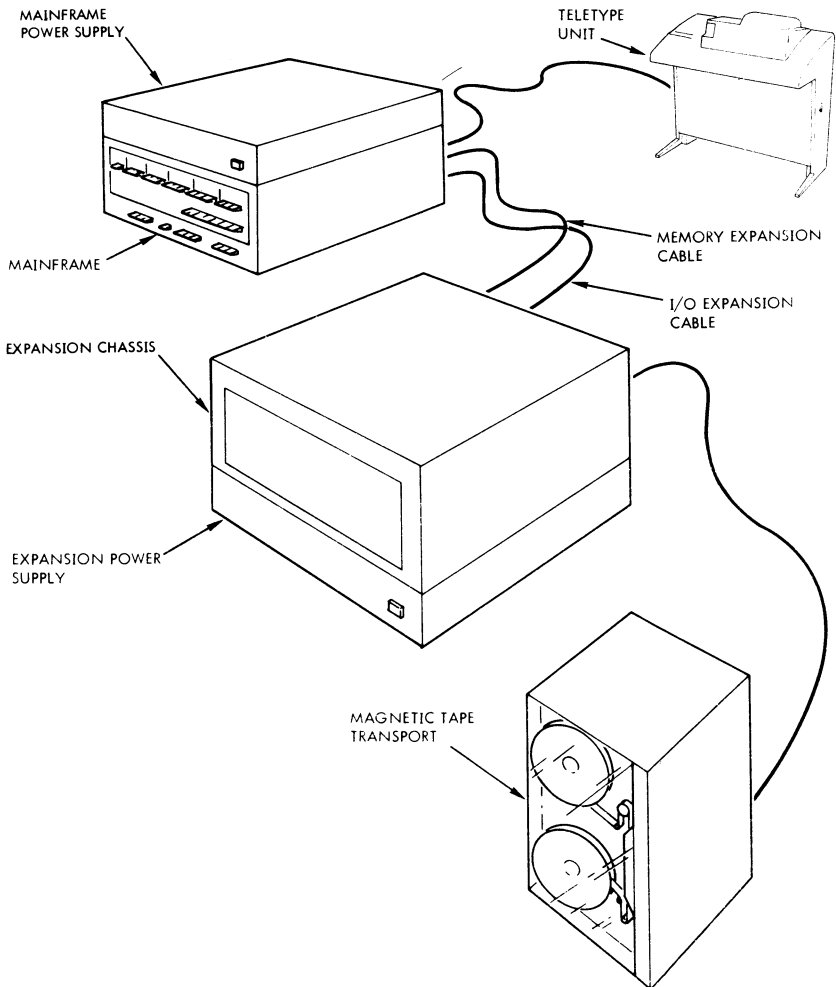
terminated with a PMA termination shoe. The PMA cable is 2.5 feet long if expansion chassis 1 is located directly under the mainframe; however, the cable can be up to 20 feet long if required (approximately 2 feet of cable are used inside expansion chassis 1).

Chassis Positions

The computer chassis can be placed side by side, back to back, or rack-mounted in a standard RETMA enclosure.

The relative position of each computer chassis is not restricted; for example, the mainframe does not have to be installed immediately adjacent to an expansion chassis.

Figure 16-12 illustrates interconnection of a typical system containing a mainframe, expansion chassis, TTY unit, and a magnetic tape transport.



VT11-0660

Figure 16-12. Typical Varian 620/f System

Signal	Pin	Signal	Pin
GRD	E1	DL6	A26
—	A1	DL11	B26
—	B1	DL8	A27
YCS	A2	DL10	B27
XCS	B2	GRD	A28
RET	A3	GRD	B28
RET	B3	—	A29
+3V	A4	—	B29
+3V	B4	—	A30
GRD	A5	—	B30
GRD	B5	MB14-	A31
DL1	A6	MD14-	B31
DL2	B6	MB15-	A32
DL0	A7	MD15-	B32
DL5	B7	MB12-	A33
DL3	A8	MD12-	B33
DL4	B8	MB13-	A34
GRD	A9	MD13-	B34
GRD	B9	—	A35
SAO1-	A10	—	B35
SAO0-	B10	MB10-	A36
SAO3-	A11	MD10-	B36
SAO2-	B11	MB11-	A37
SAO5-	A12	MD11-	B37
SAO4-	B12	MB09-	A38
GRD	A13	MD09-	B38
GRD	B13	MB08-	A39
SAO7-	A14	MD08-	B39
SAO6-	B14	MB06-	A40
SAO8-	A15	MD06-	B40
SAO9-	B15	MB07-	A41
SAO11-	A16	MD07-	B41
SAO10-	B16	—	A42
SAO13-	A17	—	B42
SAO12-	B17	MB04-	A43
GRD	A18	MD04-	B43
GRD	B18	MB05-	A44
SAO15-	A19	MD05-	B44
SAO14-	B19	MB02-	A45
—	A20	MD02-	B45
—	B20	MB03-	A46

Figure 16-13. Pin Assignments. Left Memory Expansion Cable (1 of 2)

Signal	Pin	Signal	Pin
DL15	A21	MD03-	B46
—	B21	MB00-	A47
DL12	A22	MD00-	B47
—	B22	MB01-	A48
GRD	A23	MD01-	B48
GRD	B23	—	A49
DL13	A24	—	B49
DL14	B24	—	A50
DL7	A25	—	B50
DL9	B25		

Figure 16-13. Pin Assignments, Left Memory Expansion Cable (2 of 2)

Signal	Pin	Signal	Pin
GRD	E1	YA0-	A26
—	A1	YA1-	B26
—	B1	YA2-	A27
—	A2	YA3-	B27
—	B2	YB0-	A28
ABS15-	A3	YB1-	B28
AB309-	B3	GRD	A29
ABS14-	A4	GRD	B29
ABS10-	B4	YB2-	A30
ABS13-	A5	YB3-	B30
ABS07-	B5	XA9-	A31
ABS12-	A6	XA1-	B31
ABS11-	B6	XA2-	A32
—	A7	XA3-	B32
—	B7	GRD	A33
—	A8	GRD	B33
ABS08-	B8	XB0-	A34
—	A9	XB2-	B34
ABS06-	B9	XB1-	A35
—	A10	XB3-	B35
ABS05-	B10	XB7-	A36
MBZY	A11	XB6-	B36
ABS04-	B11	GRD	A37
—	A12	GRD	B37
—	B12	XB5-	A38
PFF	A13	XB4-	B38
ABS02-	B13	XA7-	A39
SC-	A14	XA6-	B39

Figure 16-14. Pin Assignments, Right Memory Expansion Cable (1 of 2)

ABS01-	B14	XA5-	A40
MDR-	A15	XA4-	B40
ABS00-	B15	GRD	A41
MST-	A16	GRD	B41
ABS03-	B16	YB7-	A42
GRD	A17	YB6-	B42
GRD	B17	YB5-	A43
GRD	A18	YB4-	B43
GRD	B18	YA7-	A44
GRD	A19	YA6-	B44
PXWT-	B19	GRD	A45
INHT-	A20	GRD	B45
PXRT-	B20	YA5-	A46
GRD	A21	YA4-	B46
GRD	B21	SS2-	A47
YRT-	A22	SS3-	B47
YWT-	B22	SS1-	A48
SAS2-	A23	SS0-	B48
SAS1-	B23	GRD	A49
ARB12-	A24	GRD	B49
—	B24	+3V	A50
GRD	A25	+3V	B50
GRD	B25		

Figure 16-14. Pin Assignments, Right Memory Expansion Cable (2 of 2)

Signal	Pin	Signal	Pin
—	A1	GRD	A26
+3V	B1	PR4X-I	B26
—	A2	GRD	A27
+3V	B2	PR3X-I	B27
GRD	A3	GRD	A28
PR5X-I	B3	PR2X-I	B28
GRD	A4	GRD	A29
PR6X-I	B4	PR1X-I	B29

Figure 16-15. Pin Assignments, I/O Expansion Cable (1 of 2)

GRD	A5	GRD	A30
PR7X-I	B5	TPOX-I	B30
GRD	A6	GRD	A31
PR8X-I	B6	TPIX-I	B31
GRD	A7	GRD	A32
PR9X-I	B7	SERX-I	B32
GRD	A8	GRD	A33
PR10X-I	B8	DRYX-I	B33
GRD	A9	GRD	A34
IL01-	B9	FRYX-I	B34
GRD	A10	GRD	A35
IL06-	B10	EB15-I	B35
GRD	A11	GRD	A36
IL03-	B11	EB14-I	B36
GRD	A12	GRD	A37
IL00-	B12	EB13-I	B37
GRD	A13	GRD	A38
IL02-	B13	EB12-I	B38
GRD	A14	GRD	A39
IL04-	B14	EB11-I	B39
GRD	A15	GRD	A40
IL05-	B15	EB10-I	B40
GRD	A16	GRD	A41
IL07-	B16	EB09-I	B41
GRD	A17	GRD	A42
SPARE #1	B17	EB08-I	B42
GRD	A18	GRD	A43
SPARE #2	B18	EB07-I	B43
GRD	A19	GRD	A44
SPARE #3	B19	EB06-I	B44
GRD	A20	GRD	A45
SPARE #4	B20	EB05-I	B45
GRD	A21	GRD	A46
IUJX-I	B21	EB04-I	B46
GRD	A22	GRD	A47
IURX-I	B22	EB03-I	B47
GRD	A23	GRD	A48
IUCX-I	B23	EB02-I	B48
GRD	A24	GRD	A49
IUZX-I	B24	EB01-I	B49
GRD	A25	GRD	A50
SYRT-I	B25	EB00-I	B50

Figure 16-15. Pin Assignments, I/O Expansion Cable (2 of 2)

Signal	Pin	Signal	Pin
-20V	A	Com Sense	M
-20V	B	PFA Ret	N
-20V	C	PFA	P
+12V	D	-20V	R
+12V	E	-20V	S
-5V	F	-5V	T
PFF	G	PFF Ret	U
+40V	H	Therm Ret	V
+40V	J	Sense Ret	W
Term In	K	-5V	X
+3V	L		

Figure 16-16. Pin Assignments, Power Supply to Mainframe Cable

Signal	Pin	Signal	Pin
-20V	A	Com Sense	M
-20V	B	PFA Ret	N
-20V	C	PFA	P
+12V	D	-20V	R
+12V	E	-20V	S
-5V	F	-5V	T
PFF	G	PFF Ret	U
+40V	H	PFA Ret	V
+40V	J	Sense Ret	W
Therm In	K	-5V	X
+3V	L		

Figure 16-17. Pin Assignments, Memory Expansion Power Cable

Signal	Pin	Signal	Pin
+5V	A	+5V Sense	M
+5V	B	+5V	N
-12V	C	+5V	P
-12V	D	+5V	R
+12V	E	-12V	S
+1 2V	F	+12V	T
-5V	G	-5V	U
-5V	H	GRD	V
GRD	J	GRD	W
GRD	K	GRD	X
GRD	L		

Figure 16-18. Pin Assignments, I/O Expansion Power Cable

Signal	Pin
115VAC	A
115V Ret	B
24VAC	C
RTC Ret	D
Relay Coil Out	E
GRD	F

Figure 16-19. Pin Assignments, AC Power Cable

Signal	Pin	Signal	Pin
GRD	A	IL04-	K
GRD	B	IL05-	L
IL01-	C	GRD	M
IL06-	D	IL07-	N
IL03-	E	GRD	P
GRD	F	GRD	R
GRD	G	IL00-	S
+5V	H	GRD	T
IL02-	J		

Figure 16-20. Pin Assignments, PIM Cable

Signal	Pin	Signal	Pin
TAKX- Ret	A	TROX-B	K
—	B	TAKX-B	L
DESX-B	C	TROX- Ret	M
CDCX-B	D	—	N
BCDX-B	E	DESX- Ret	P
BCDX- Ret	F	CDCX- Ret	R
DCEX- Ret	G	DCEX-B	S
—	H	TRQX- Ret	T

Figure 16-21. Pin Assignments, BIC Cable

Signal	Pin	Signal	Pin
GRD	A1 to A50	PD06 +	B26
PA14 +	B1	PD05 +	B27
PA13 +	B2	PD04 +	B28
PA12 +	B3	PD03 +	B29
PA11 +	B4	PD02 +	B30
PA10 +	B5	PD01 +	B31
PA09 +	B6	PD00 +	B32
PA08 +	B7	—	B33
PA07 +	B8	READ-	B34
PA06 +	B9	GO-	B35
PA05 +	B10	—	B36
PA04 +	B11	ACK1-	B37
PA03 +	B12	ACK2-	B38
PA02 +	B13	ACK3-	B39
PA01 +	B14	ACK4-	B40
PA00 +	B15	PMRS-	B41
—	B16	REQ1-	B42
PD15 +	B17	REQ2-	B43
PD14 +	B18	REQ3-	B44
PD13 +	B19	REQ4-	B45
PD12 +	B20	—	B46
PD11 +	B21	SYRT-P	B47
PD10 +	B22	+3V	B48
PD09 +	B23	+3V	B49
PD08 +	B24	+3V	B50
PD07 +	B25		

Figure 16-22. Pin Assignments, PMA Cable

SECTION 17 – MAINTENANCE

Integrated-circuit design reduces the occurrence of malfunctions in Varian 620/f systems. When problems do occur, however, diagnostic programs and troubleshooting procedures can pinpoint the fault and quickly return the computer to service.

The power failure/restart (PF/R) option provides an orderly shutdown in case of power failure or turn-off and restarts the program when power is restored.

To prevent accidental setting of control panel controls during computer operation, all switches can be disabled with the power switch.

Routine Maintenance

In general, the Varian 620/f system requires no routine maintenance other than a periodic check of timing waveforms in the memory. If necessary, adjustments should be performed in accordance with procedures outlined in the instruction manual for the memory.

Cooling fans in the memory, the power supplies, and the equipment interfaces manufactured by Varian Data Machines, are permanently lubricated and need no routine attention. Routine maintenance procedures for attached electro-mechanical devices are contained in the manufacturer's instruction manuals furnished with the equipment.

Diagnostic Programs

Diagnostic programs (see Section 26) locate malfunctions in a system (troubleshooting) or aid in locating mistakes in a computer

program (debugging). These programs are used off-line when the computer is not performing data transfers or control functions.

The diagnostic programs, which are on punched paper tape, exercise the computer and associated peripheral devices with sequences of instructions. The results of these sequences are checked against the proper responses. If an instruction or other operation is improperly executed, the program ends the sequence and causes the printing of information indicating which instruction or operation failed. One can then repeat, continue, or halt the diagnostic routine until the fault is isolated and corrected.

Troubleshooting

The time required to correct system malfunctions will depend upon the efficiency of the procedures used. Although various specified techniques may be suggested, there is no real substitute for an ordered, logical analysis of the problem and isolation of the cause to the level of a replaceable component. Such an analysis can be based only upon knowledge of the equipment design. The discussion of troubleshooting procedures first describes general techniques and then specific procedures recommended for the Varian 620/f.

The condition of the machine can be determined by observing the diagnostic list-outs and the display indicators on the console. The displays should be observed while executing the system in the STEP mode. These simple procedures generally reveal the

faulty functional area. Continued operation, using REPEAT where applicable, and oscilloscope use will help locate the faulty circuit board or other replaceable element. When the faulty element is isolated, it may be replaced and the machine returned to operation. The faulty component on the circuit board or other element may be located and replaced off-line.

The general instructions for troubleshooting are as follows (see Figure 17-1):

- a. Analyze the problem
- b. Look for obvious solutions first
- c. Isolate problem to a functional area
- d. Analyze fault within the problem area
- e. Correct the fault
- f. Restore conditions for normal operation

Analyzing the Problem

Take time to thoroughly analyze the problem. For example, if the ADD instruction does not produce correct results, is the fault a function of the sign (plus or minus), of the carries, or of some other element? Are the operational registers being properly loaded? Use the displays for gathering the necessary data.

Seeking Obvious Solutions

Make sure that a malfunction has actually occurred. Relate problems to recent events such as cleaning or servicing. Look for improperly set controls or test equipment and accidental disconnections of plugs, etc. Consider miscellaneous temporary failures, such as mechanical jamming of peripheral equipments.

Isolating to a Functional Area

Isolate the fault to a functional area: that is, memory, control, arithmetic/logic, opera-

tional register, input/output, peripheral controller, or peripheral device. The process of isolating the malfunctioning area is generally a straightforward process of eliminating the functional areas which are operating properly, when the faulty area is not immediately obvious.

Analyzing the Area

When the fault has been isolated to a functional area, the rest of the system may be temporarily ignored. Use the logic and timing diagrams, and observe circuit waveforms to quickly isolate the problem to an individual replaceable element.

Correcting the Fault

Replace the faulty circuit board or other element. Attempt to analyze the cause of the failure before restoring power to prevent the possibility of the failure occurring again.

Restoring Normal Operation

When the system is again operating properly, make sure it is returned to normal operating conditions. If circuit boards have been unseated, be sure all are properly reseated in the connectors. Set all panel controls, close all circuit breakers, insure that fans are running, etc. Finally, verify proper operation by running the diagnostic tests.

Varian 620/f Troubleshooting Procedures

The checkout procedures outlined in the following paragraphs may be used when it has been determined that the computer is not operating properly. This is determined, typically, by various programmed maintenance routines. A typical computer failure (if in the central processor) will produce failures in a large percentage of routines. For this reason, determining a computer

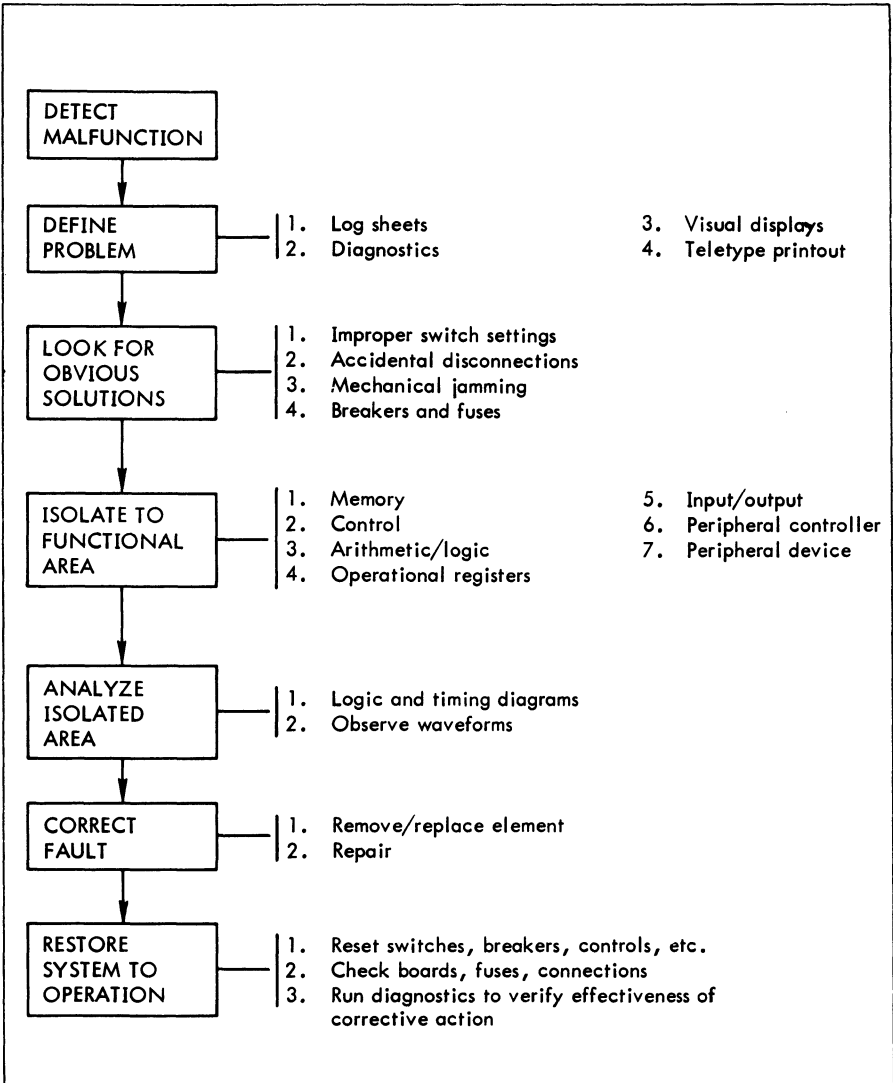


Figure 17-1. General Troubleshooting Steps

maintenance

failure is quite simple. Memory failures of a single bit or word are quite rare and will generally require special test procedures which may result in clearing the section of memory involved.

The four major sections of the computer for troubleshooting purposes are as follows:

- a. Arithmetic and Register Section
- b. Control, Timing, and Decoding Section
- c. Memory Section
- d. I/O Section

The general characteristics of the failure will, in many cases, immediately indicate the failed section. If the failure is catastrophic and all instructions fail, the cause is usually in the timing section. A failure in the Arithmetic or Register section will normally be evidenced by the failure to perform arithmetic operations or to increment the program counter correctly. Memory failures may be indicated by repeated occurrence of the halt instruction, or complete random sequencing of instructions. I/O failures are restricted to I/O instructions and are associated only with the logic of the I/O device. An I/O failure is easily recognized by noting failures in various I/O routines, but not strictly internal routines.

A general procedure for locating the basic problem area is outlined as follows:

- a. Set "1's" into all bit positions of the operational registers A, B, X, and P, and into all bit positions of the Instruction Register (I), by means of the switches on the control console. Then reset all registers to zero. This operation checks basic set and reset functions for all register flip-flop circuits in the computer. The operation does not, however, check gating into these registers from the internal busses.
- b. Set all "1's" into the Accumulator Register (A). Single-step STA to location 0, then single-step LDA from location 0 and check for agreement. Next implement an Increment Memory and Replace instruction. Finally, increment a location that is all "0's".
- c. Set Instruction Register to Increment A and put into Repeat Mode. Set A Register to all "1's" except low-order bit. Press STEP switch twice and note results. On the second step, A should go to "0". This provides a major check of the Arithmetic Unit.

Equipment	Description
Oscilloscope	Tektronix, type 547 (or equivalent) with dual-trace plug-in unit
Multimeter	Simpson 260 (or equivalent)
Card Extenders	a. Extender cards for cards in CPU tray b. Extender cards for peripheral controllers
Extension Cable	Required for CPU and memory trays
Card Puller	Tichener 1731
Wire-Wrap Gun	Gardener-Denver 14R2 (or equivalent)
Soldering Iron	39-watt pencil type

Figure 17-2. Recommended Test Equipment

SECTION 18 – DATA AND INSTRUCTION FORMATS

There are two basic word formats used in the Varian 620/f: data and instruction.

The instruction word format is further divided into four types: single-word addressing, single-word non-addressing, double-word addressing and double-word non-addressing.

Data Word Formats

Data words may contain operands, operand addresses or indirect addresses, depending upon the instruction or addressing mode in process.

Data Words

The data word format is shown in Figure 18-1.

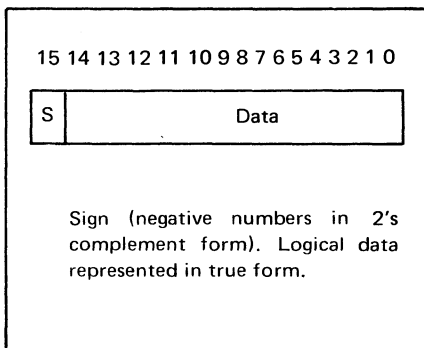


Figure 18-1. Data Word Format

Data bits occupy positions 0-14, with the sign in position 15. Negative numbers are represented in 2's complement form.

Indirect Addresses

When the data word is an indirect address, rather than an operand, it has the format shown in Figure 18-2.

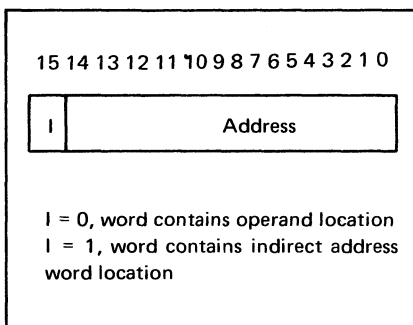


Figure 18-2. Indirect-Address Data Word Format

An indirect-address word is accessed by an instruction that is in the indirect address mode (see Section 19).

Bit 15 contains the I bit, which designates whether the memory location that is addressed by the indirect-address word contains the operand (I=0), or contains the location of yet another indirect address word (I=1).

Indirect addressing may be extended to any desired level (see description of Memory Protect option, Section 10, for exception). Each level of indirect addressing adds one cycle (750 nanoseconds) to the basic execution time of an instruction.

Instruction Word Formats

Instruction words may be either addressing or non-addressing, single-word or double-word.

The basic instruction word format is shown in Figure 18-3.

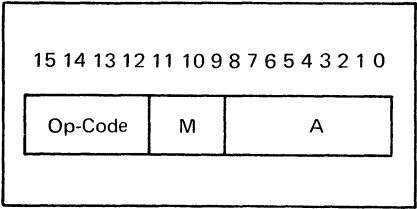


Figure 18-3. Instruction Word Format (Single-Word Instructions and First Word of Double-Word Instructions)

The format shown is applicable to all instruction words. For double-word instructions, the format shown applies to the first instruction word.

The instruction word is divided into three fields; op-code field, M field and A field. The function of the three fields vary according to the type of instruction, but may generally be defined as follows:

Op-code field	bits 12-15	Designates type of instruction (e.g., single-word addressing, I/O instructions or other).
M field	bits 9-11	Designates address mode or mode of operation
A field	bits 0-8	Contains a variety of information depending upon type of instruction

Single-Word Addressing Instructions

Instruction groups applicable to this type of instruction are the direct versions of:

- Load/Store
- Arithmetic
- Logical

These instruction groups are designated by octal number 01 through 07, and 11 through 17 in the op-code field. The M field contains one of the following addressing modes:

Direct	binary 0 X X
Relative mode	binary 1 0 0
Index (X)	binary 1 0 1
Index (B)	binary 1 1 0
Indirect	binary 1 1 1

For direct addressing, bits 9 and 10 of the M field are combined with the A field to form a direct address to any of 2048 locations.

Single-Word Non-Addressing Instructions

Instruction groups applicable to this type of instruction are:

- Shift
- Control
- Register Change
- Input/Output

The op-code field contains octal 00 except for the last type, Input/output, which is designated by octal 10. The M field designates the mode of operation, and the A field specifies the action to be performed by the computer such as:

- a. Number of shifts
- b. Kind of register change as well as source and destination registers
- c. Input/output

Double-Word Addressing Instructions

Instruction groups applicable to this type of instruction are:

Jump
Jump and Mark
Execute

Extended Address Versions of:

Load/Store
Arithmetic
Logic

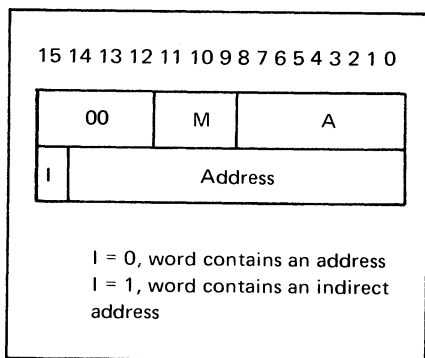


Figure 18-4. Double-Word Addressing Instruction Format (Except Extended Instructions)

The format for both the first and second words of two-word addressing instructions other than the optional extended-addressing instructions, is given in Figure 18-4. The op-code field contains octal 00; the M field an octal 1, 2, 3 or 6, designating the mode of instruction to be performed; and the A

field defines the logical states which condition the execution of the instruction.

The second word contains address of either an instruction or operand, or the location of the instruction to be executed if the condition is met. Indirect addressing is permitted.

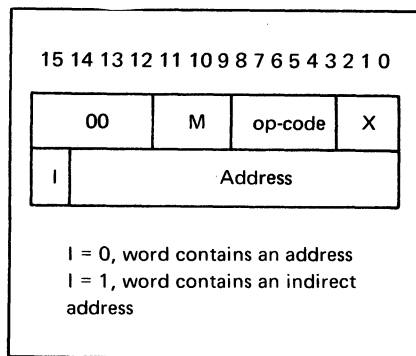


Figure 18-5. Extended Addressing Instruction Format

For the extended address instructions (see Section 19), the A field is further divided into two sub-fields. Bits 0-2 form the X field, as shown in Figure 18-5, and are coded to indicate the address mode. Bits 3-8 contain any single-word operation instruction which, in a single word instruction, ordinarily appear in the op-code field.

Double-Word Non-Address Instructions

Instruction groups applicable to this type of instruction are the immediate-addressing versions of the following:

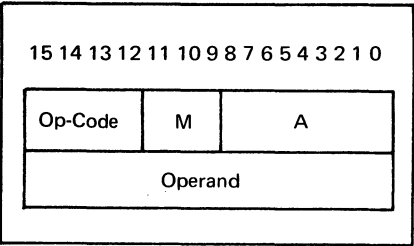
Load/Store
Arithmetic
Logical

The format for these instructions is shown in Figure 18-6.

instruction-word formats

The op-code field contains octal 00 and the M field contains an octal 6. The A field contains the operation mode (octal 0) in bits 0-2, and bits 3-8 contain the equivalent of a single word operation instruction.

Since indirect addressing is not permitted, the second word always contains an operand.



**Figure 18-6. Double-Word
Non-Addressing (Immediate)
Instruction Format**

SECTION 19 – ADDRESSING MODES

The Varian 620/f features a number of addressing modes that can be used by the programmer to increase the efficiency of his program.

The addressing mode is a function of both the type of instruction and the coding within the instruction.

Three types of instructions are involved: single-word addressing, extended, and immediate.

Figure 19-1 summarizes the relationship between the three instruction types and the eight addressing modes.

Immediate instructions are limited to the immediate-addressing mode. Single-word addressing instructions may be used in all of

the addressing modes except immediate. Extended instructions can take advantage of all eight addressing modes.

Address-mode codes for the single-word instruction and extended instructions are given in Figures 19-2 and 19-3. For additional details on the format of these instructions, see Section 18.

Immediate Addressing

Immediate Addressing with Immediate Instructions

Immediate instructions are nominally classified (Section 18) as double-word non-addressing instructions since the second word of the instruction is the operand itself. No further addressing of memory is required.

Addressing Mode	Instruction Type		
	Single-Word	Extended	Immediate
Immediate		X	X
Direct	X	X	
Indirect	X	X	
Preindexed with X Register	X	X	
Postindexed with X Register		X	
Preindexed with B Register	X	X	
Postindexed with B Register		X	
Relative to P Register	X		

Figure 19-1. Relationship Between Instruction Type and Addressing Mode

"Immediate addressing" is included in this discussion, however, because it is one of the options available to the programmer for accessing operands stored in memory.

The address of the operand is, in this case, the memory location containing the second word of the instruction. The CPU addresses this location when it fetches the immediate instruction for execution.

Since there is no separate addressing phase, no modification of the address is possible. Indexed, relative, and indirect addressing do not apply to immediate instructions.

Immediate Addressing with Extended Instructions

Extended-address instructions may be used in the immediate mode by inserting an octal 0, 1, 2, or 3 in the X field (See Figures 18-5 and 19-3) of the first word of the instruction.

With this code in effect, the CPU processes the second word of the instruction as an operand rather than as an address.

Direct Addressing

In direct addressing, the address of the operand is contained within the instruction itself.

Direct Addressing with Single-Word Instructions

Single-word addressing instructions operate in the direct mode whenever the most significant bit of the M field (see Figures 18-3 and 19-2) is a 0.

The remaining two bits of the M field are combined with the nine bits in the A field to form an 11-bit effective address. The address directs the CPU to an operand

stored in the first 2048 words (0000 thru 2047) of memory.

Direct Addressing with Extended Instructions

Extended-addressing instructions operate in the direct mode whenever an octal 7 is inserted in the X field and the most significant bit of the second word is 0.

The remaining 15 bits of the second word form the effective address of the operand. Any location in a full 32K of memory may be directly addressed.

Indirect Addressing

In indirect addressing, the address of the operand is stored in memory at a location specified by the instruction.

Indirect Addressing with Single-Word Instructions

Single-word addressing instructions operate in the indirect mode whenever an octal 7 is inserted in the M field.

The 9 bits of the A field direct the CPU to an address location in the first 512 words of memory. The word stored in that location is the address of the operand.

Indirect Addressing with Extended Instructions

Extended-addressing instructions operate in the indirect mode whenever the most significant bit of the second word is 1.

The remaining 15 bits of the second word direct the CPU to any address location in a full 32K of memory. The word stored in that location is the address of the operand.

Multi-Level Indirect Addressing

The word stored in the memory location specified by any indirect-addressing instruction may itself be an indirect address. The CPU is directed to this second memory location to determine the address of the operand.

Any number of indirect addressing levels is permitted. Each level adds one machine cycle (750 nanoseconds) to the time required for the execution of the instruction. (See description of Memory Protect option, Section 10, for exceptions.)

Indexed with X Register

The effective address may be modified by adding it to the contents of the X register.

X-Register Indexing with Single-Word Instructions

Single-word addressing instructions may be indexed with the X register by inserting an octal 5 in the M field.

The contents of the A field are added to the contents of the X register to form the effective address of the operand.

X-Register Preindexing and Postindexing with Extended Instructions

Extended-addressing instructions may be indexed with the X register by inserting an octal 5 in the X field of the first word of the instruction. (See Figure 19-4 for effect of pre- and postindexing.)

If bit 7 of the first word is zero, preindexing is specified. The contents of the second word of the instruction are added to the contents of the X register, and the result is used as intermediate address for subsequent indirect-addressing steps.

If bit 7 of the first word is one, postindexing is specified. All indirect-addressing steps are performed first, and the final indirect address is then added to the contents of the X register to form the effective address.

Indexed with B Register

The effective address may be modified by adding it to the contents of the B register. contents of the B register.

B-Register Indexing with Single-Word Instructions

Single-word addressing instructions may be indexed with the B register by inserting an octal 6 in the M field.

The contents of the A field are added to the contents of the B register to form the effective address of the operand.

B-Register Preindexing and Postindexing with Extended Instructions

Extended-addressing instructions may be indexed with the B register by inserting an octal 6 in the X field of the first word of the instruction.

If bit 7 of the first word is zero, preindexing is specified. The contents of the second word of the instruction are added to the contents of the B register, and the result is used as intermediate address for subsequent indirect-addressing steps.

If bit 7 of the first word is one, postindexing is specified. All indirect-addressing steps are performed first, and the final indirect address is then added to the contents of the B register to form the effective address.

Relative Addressing

In relative addressing, the address contained

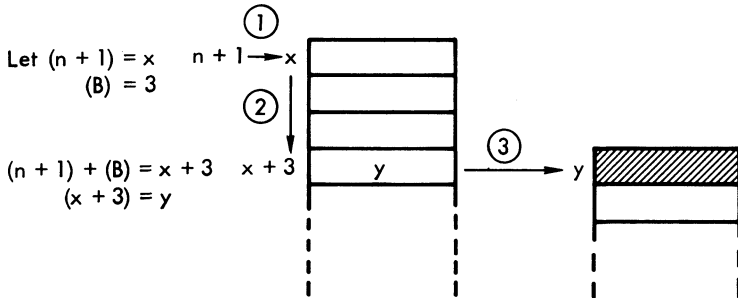
M Field (octal)	Addressing Mode	Operation
0-3	Direct	Two least significant bits of M field are combined with A field to form effective address.
4	Relative	A field is added to contents of P register plus one to form effective address.
5	Indexed with X register	A field is added to contents of X register to form effective address.
6	Indexed with B register	A field is added to contents of B register to form effective address.
7	Indirect	A field specifies location in which address of operand is stored.

Figure 19-2. Address Coding for Single-Word Instructions

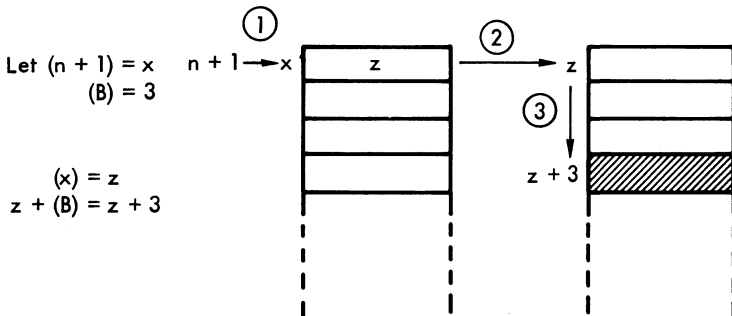
X Field (octal)	Addressing Mode	Operation
0-3	Immediate	Second word of instruction contains the operand.
4	Relative	Second word of instruction is added to contents of P register plus one to form effective address.
5	Indexed with X register	Second word of instruction is added to contents of X register to form effective address.
6	Indexed with B register	Second word of instruction is added to contents of B register to form effective address.
7	Direct	If the most significant bit of the second word is 0, the remaining bits are the operand address.
7	Indirect	If the most significant bit of the second word is 1, the remaining bits identify the location of the operand address

Figure 19-3. Address Coding for Extended-Addressing Instructions

PREINDEXING



POSTINDEXING



Circled numbers indicate the sequence of events.
 Shaded area is the effective memory address.

Figure 19-4. Preindexing and Postindexing

addressing modes

within an instruction is modified by adding it to the contents of the P register plus one.

Relative Addressing with Single-Word Instructions

Single-word addressing instructions operate in the relative mode whenever an octal 4 is inserted in the M field.

The contents of the A field are added to the contents of the P register to form the effective address of the operand. This permits addressing locations up to 512 words in advance of the current program locations.

NOTE: P register is +1 from the instruction.

Relative Addressing with Extended Instructions

Extended-addressing instructions operate in the relative mode whenever an octal 4 is inserted in the X field of the first word of the instruction.

The contents of the second word of the instruction are added to the contents of the P register to form the effective address of the operand.

SECTION 20 – INSTRUCTION SET

The Varian 620/f central processor is capable of decoding and executing over 100 different instructions.

This extensive instruction repertoire allows the programmer to write programs that make efficient use of machine time and minimize the space required in memory for program storage.

The instructions can be divided into three functional groups:

- a. Instructions that involve an interaction between the central processor and memory.
- b. Instructions that relate to operations within the central processor itself.
- c. Instructions that control input-output operations.

The following pages describe the instructions in detail and are arranged in groups, according to function.

Memory/CPU Instructions	Page
Load/Store	20-2
Arithmetic	20-5
Logical	20-11
Jump	20-15
Jump & Mark	20-19
Execution	20-23
Internal CPU Instructions	
Control	20-27
Shift	20-29
Register Change	20-33

I/O Instructions

Sense	20-40
External Control	20-40
I/O, Registers	20-41
I/O, Memory	20-43

The instruction descriptions are followed (page 20-45) by flow diagrams showing how the various types of instructions are processed. These are followed, in turn, by alphabetical and numerical listings of all Varian 620/f instructions.

NOTE: For an explanation of the M, A, X and I fields shown in the following descriptions, see Sections 18 and 19.

Load/Store Instructions

This group comprises the instructions for loading registers from memory or for storing the contents of registers in memory. Subgroups of these instructions permit such loading or storing in normal, extended, or immediate addressing modes.

Within each load/store instruction, one octal group specifies the operation as follows:

001	Loads the A register
010	Loads the B register
011	Loads the X register
101	Stores the A register
110	Stores the B register
111	Stores the X register

This specification is in bits 12-14 of one-word, and bits 3-5 (of the first word) of two-word load/store instructions

load/store instructions

LDA

Load A Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

01	M	A
----	---	---

The contents of the effective memory location are placed in the A register.

Timing: 2 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A

LDB

Load B Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

02	M	A
----	---	---

The contents of the effective memory location are placed in the B register.

Timing: 2 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: B

LDAI

Load A Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	010
Operand		

The contents of the operand at location $n + 1$ are placed in the A register.

Timing: 2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: A

LDBI

Load B Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	020
Operand		

The contents of the operand at location $n + 1$ are placed in the B register.

Timing: 2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: B

LDAE

Load A Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	01	X
I	Operand Address		

The contents of the memory location, as addressed by the operand address at location $n + 1$, are placed in the A register.

Timing: 3 cycles
Indexing: Yes
Indirect Addressing: Yes
Register Altered: A

LDBE

Load B Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		6	02	X
1	Operand Address			

The contents of the memory location as addressed by the operand address at location $n + 1$ are placed in the B register.

Timing: 3 cycles
Indexing: Yes
Indirect Addressing: Yes
Register Altered: B

LDX Load X Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

03	M	A
----	---	---

The contents of the effective memory location are placed in the X register.

Timing: 2 cycles
 Relative: Yes
 Indexing: Yes
 Indirect Addressing: Yes
 Registers Altered: X

LDXI Load X Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	030
Operand		

The contents of the operand at location $n + 1$ are placed in the X register.

Timing: 2 cycles
 Indexing: No
 Indirect Addressing: No
 Registers Altered: X

LDXE Load X Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	03	X
I	Operand Address		

The contents of the memory location as addressed by the operand address at location $n + 1$ are placed in the X register.

Timing: 3 cycles
 Indexing: Yes
 Indirect Addressing: Yes
 Register Altered: X

STA Store A Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

05	M	A
----	---	---

The contents of the A register are placed in the effective memory location.

Timing: 2 cycles
 Relative: Yes
 Indexing: Yes
 Indirect Addressing: Yes
 Registers Altered: Memory

STAI Store A Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	050
Operand		

The contents of the A register are placed in the operand at location $n + 1$.

Timing: 2 cycles
 Indexing: No
 Indirect Addressing: No
 Registers Altered: Operand

STAE Store A Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	05	X
I	Operand Address		

The contents of the A register are stored in the memory location as addressed by the operand address at location $n + 1$.

Timing: 3 cycles
 Indexing: Yes
 Indirect Addressing: Yes
 Register Altered: Memory

load/store instructions

STB

Store B Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

06	M	A
----	---	---

The contents of the B register are placed in the effective memory location.

Timing: 2 cycles
 Relative: Yes
 Indexing: Yes
 Indirect Addressing: Yes
 Registers Altered: Memory

STX

Store X Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

07	M	A
----	---	---

The contents of the X register are placed in the effective memory location.

Timing: 2 cycles
 Relative: Yes
 Indexing: Yes
 Indirect Addressing: Yes
 Registers Altered: Memory

STBI

Store B Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	060
Operand		

The contents of the B register are placed in the operand at location $n + 1$.

Timing: 2 cycles
 Indexing: No
 Indirect Addressing: No
 Registers Altered: Operand

STXI

Store X Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	070
Operand		

The contents of the index register are placed in the operand at location $n + 1$.

Timing: 2 cycles
 Indexing: No
 Indirect Addressing: No
 Registers Altered: Operand

STBE

Store B Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	06	X
I	Operand Address		

The contents of the B register are stored in the memory location as addressed by the operand address at location $n + 1$.

Timing: 3 cycles
 Indexing: Yes
 Indirect Addressing: Yes
 Register Altered: Memory

STXE

Store X Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	07	X
I	Operand Address		

The contents of the X register are stored in the memory location as addressed by the operand address at location $n + 1$.

Timing: 3 cycles
 Indexing: Yes
 Indirect Addressing: Yes
 Register Altered: Memory

TSA**Transfer Switches
to A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	7	402
----	---	-----

The contents of the Load/Entry front panel switches are transferred to the A register.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A

Arithmetic Instructions

This group comprises the instructions for incrementation of the contents of a memory address; for performing the arithmetic functions of addition, subtraction, multiplication, and division; and the extended-and immediate-addressing counterparts of these instructions. Multiplication and division are optional.

In the one-word instructions, bits 12-15 specify the arithmetic function as follows:

0	100	Increment
1	010	Add
1	100	Subtract
1	110	Multiply
1	111	Divide

In the two-word instructions, the same configurations appear in bits 3-6 of the first word.

INR Increment Memory and Replace

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

04	M	A
----	---	---

The contents of the effective memory locations are incremented by one.

When the value of memory is 077777 (maximum positive number) and INR is executed to memory, overflow (OF) is set and the value of memory is 100000.

Timing: 3 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: Memory, OF

INRI Increment and Replace Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	040
Operand		

The contents of the operand at location $n + 1$ are incremented by one. When the value

of memory is 077777 (maximum positive number) and INRI is executed to memory, OF is set and the value of memory is 100000.

Timing: 3 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: Operand, OF

INRE Increment Memory and Replace Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	04	X
I	Operand Address		

The contents of the memory location as addressed by the operand address at location $n + 1$ are incremented by one. When the value of memory is 077777 (maximum positive number) and INRE is executed to memory, OF is set and the value of memory is 100000.

Timing: 4 cycles
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: Memory, OF

ADD Add Memory to A Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

12	M	A
----	---	---

The contents of the effective memory location are added to the contents of the A register and the sum is placed in the A register.

If the value of the A register is 077777 (maximum positive number) and any nonzero positive number (n) is added to it, OF is set and the sign (bit 15) is set to one. The contents of bits 0 to 14 will be n-1.

For example, adding 000002 to 077777 results in OF being set to one and the A register containing 100001 (octal).

Timing: 2 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A, OF

ADDI Add to A Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	120
Operand		

The contents of the A register are added to the contents of the operand at location n + 1. The sum is placed in the A register.

If the value of the A register is 077777 (maximum positive number) and any nonzero positive number (n) is added to it,

OF is set and the sign (bit 15) is set to one. The contents of bits 0 to 14 are n-1.

For example, adding 000002 to 077777 results in OF being set to one and the A register containing number 100001 (octal).

Timing: 2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: A, OF

ADDE Add Memory to A Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	12	X
I	Operand Address		

The contents of the memory location as addressed by the operand address at location n + 1 are added to the contents of the A register and the sum is placed in the A register.

If the value of the A register is 077777 (maximum positive number) and any nonzero positive number (n) is added to it, OF is set and the sign (bit 15) is set to 1. The contents of bits 0-14 are n-1.

For example, adding 000002 to 077777 results in overflow being set to one and the A register containing 100001 (octal).

Timing: 3 cycles
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A, OF

SUB **Subtract Memory**
From A Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

14	M	A
----	---	---

The contents of the effective memory location are subtracted from the A register and the difference is placed in the A register.

If the value of the A register is 100000 (maximum negative number) and any non-zero positive number (n) is subtracted from it, OF is set and the sign (bit 15) is reset. The contents of bits 0 to 14 are the maximum positive number, (077777) less n-1.

For example, subtracting +3 from 100000 results in OF being set and the A register containing 077775.

- Timing: 2 cycles
- Relative: Yes
- Indexing: Yes
- Indirect Addressing: Yes
- Registers Altered: A, OF

SUBI **Subtract from A**
Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	140
Operand		

The contents of the operand at location n + 1 are subtracted from the contents of the A register. The difference is placed in the A register.

If the value of the A register is 100000 (maximum negative number) and any nonzero positive number (n) is subtracted

from it, OF is set and the sign (bit 15) is reset. The contents of bits 0 to 14 are the maximum positive number (077777) less n-1.

For example, subtracting +3 from 100000 results in OF being set and the A register containing 077775.

- Timing: 2 cycles
- Indexing: No
- Indirect Addressing: No
- Registers Altered: A, OF

SUBE **Subtract Memory from**
A Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	14	X
I	Operand Address		

The contents of the memory location as addressed by the operand address at location n + 1 are subtracted from the contents of the A register and the difference is placed in the A register.

If the value of the A register is 100000 (maximum negative number) and any nonzero positive number (n) is subtracted from it, OF is set and the sign (bit 15) is reset. The contents of bits 0 to 14 are the maximum positive number (077777) less n-1.

For example, subtracting +3 from 100000 results in OF being set and the A register obtaining 077775.

- Timing: 3 cycles
- Indexing: Yes
- Indirect Addressing: Yes
- Registers Altered: A, OF

MUL*

Multiply

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

16	M	A
----	---	---

The contents of the B register are multiplied by the contents of the effective memory location. The original contents of the A register are added to the product. The product is placed in the A and B registers, with the most significant half of the product in the A register and the least significant half in the B register. The sign of the product is contained in the sign position of the A register. The sign position of the B register is reset to zero. If the contents of B register and memory contain the largest possible negative number, the result will be zero and OF indicator will be set.

The algorithm is in the form (M). $B + A \rightarrow A, B$

Timing: 8.5 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A, B, OF

MULI*

Multiply Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	160
Operand		

The contents of the B register are multiplied by the contents of the operand at location n + 1 and the contents of the A register are added to the product. The result is placed in the A and B registers, with the most significant half of the product in the A register and the least significant half in the B register. The sign of the product is

contained in the sign position of the A register. The sign position of the B register is reset to zero. If the B register and memory contain the largest possible negative number, the result is zero and the OF indicator set.

The algorithm is in the form

(M). $B + A \rightarrow A, B$

Timing: 8.5 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: A, B, OF

MULE*

Multiply Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	16	X
I	Operand Address		

The contents of the B register are multiplied by the contents of the memory location as addressed by the operand address in location n + 1, and the contents of the A register are added to the product. The result is placed in the A and B registers with the most significant half of the product in the A register and the least significant half in the B register.

The sign of the product is contained in the sign position of the A register. The sign position of the B register is reset to zero. If the B register and memory contain the largest possible negative number, the result will be zero and the OF indicator set.

The algorithm is in the form

(M). $B + A \rightarrow A, B$

Timing: 9.5 cycles
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A, B, OF *optional

DIV*

Divide

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

17	M	A
----	---	---

The contents of the A and B registers are divided by the contents of the effective memory location. The quotient is placed in the B register with the sign, and the remainder is placed in the A register with the sign of the dividend.

If $(A,B)/M < 1$ (divisor $>$ dividend, taken as a binary fraction), overflow will not occur. If overflow does occur, the OF indicator is set; this results in the quotient being invalid and unpredictable.

Timing: 8.5 cycles

Relative: Yes

Indexing: Yes

Indirect Addressing: Yes

Registers Altered: A, B, OF

If $(A,B)/M < 1$ (divisor $>$ dividend, taken as a binary fraction), overflow will not occur. If overflow does occur, the OF indicator is set; this results in the quotient being invalid and unpredictable.

Timing: 8.5 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: A, B, OF

DIVI*

Divide Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	170
Operand		

The contents of the A and B registers are divided by the contents of the operand at location $n + 1$. The quotient is placed in the B register with sign, and the remainder is placed in the A register with the sign of the dividend.

DIVE*

Divide Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	17	X
I	Operand Address		

The contents of the A and B registers are divided by the contents of the memory location as addressed by the operand address at location $n + 1$. The quotient is placed in the B register and the remainder is placed in the A register.

If $(A,B)/M < 1$ (divisor $>$ dividend, taken as a binary fraction), overflow will not occur. If overflow does occur, the OF indicator is set; this results in the quotient being invalid and unpredictable.

Timing: 9.5 cycles

Indexing: Yes

Indirect Addressing: Yes

Register Altered: A, B, OF ***optional**

Logic Instructions

This group comprises the inclusive-OR, exclusive-OR, and AND instructions and their immediate-addressing and extended-addressing counterparts.

In the one-word instructions, bits 12-15 specify the logic function as follows:

1	001	Inclusive-OR
1	011	Exclusive-OR
1	101	AND

In the two-word instructions, the same configurations appear in bits 3-6 of the first word.

ORA **Inclusive-OR Memory
and A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

11	M	A
----	---	---

An inclusive-OR operation is performed between the effective memory location and the contents of the A register. The result is placed in the A register. If either the effective memory location or A register contains a one in the same bit position, a one is placed in the result. The truth table is shown below, where n = bit position.

- Timing: 2 cycles
- Relative: Yes
- Indexing: Yes
- Indirect Addressing: Yes
- Registers Altered: A

ORAI **Inclusive-OR
to A Register Immediate**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	110
Operand		

An inclusive-OR is performed between the contents of the operand and the contents of the A register. The result is placed in the A register. If either the operand or the A register contains a one in the same bit position, a one is placed in the result in the A register. The truth table is shown below, where n = bit position.

- Timing: 2 cycles
- Indexing: No
- Indirect Addressing: No
- Registers Altered: A

ORAE **Inclusive-OR Memory
and A Register
Extended**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		6	11	X
I	Operand Address			

The inclusive-OR operation is performed between the contents of the A register and the contents of the memory location as addressed by the operand address in location n + 1.

The result is placed in the A register. If either the memory or A contains a one in the same position, a one is placed in the result. The truth table is shown below, where n = bit position.

- Timing: 3 cycles
- Indexing: Yes
- Indirect Addressing: Yes
- Register Altered: A

Condition		Result
A (n)	Effective Memory Location (n)	A (n)
0	0	0
0	1	1
1	0	1
1	1	1

ERA Exclusive—OR Memory and A Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

13	M	A
----	---	---

An exclusive-OR operation is performed between the effective memory location and the contents of the A register. The result is placed in the A register. If the same bit position of the effective memory location and the A register contains a zero or if both bit positions contain a one, the result is zero. If the same bit position of the effective memory location and A are not equal (i.e., one contains a zero and the other a one), the result is a one. The truth table is shown below, where n = bit position.

Timing: 2 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A

ERAI Exclusive-OR to A Register Immediate

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	130
Operand		

An exclusive-OR is performed between the contents of the operand at location n + 1 and the contents of the A register, and the result is placed in the A register. If the same bit position of the operand and the A register contains a zero or if both bit positions contain a one, the result is zero. The truth table is shown below, where n = bit position.

Timing: 2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: A

ERAE Exclusive-OR Memory and A Register Extended

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	13	X
I	Operand Address		

An exclusive-OR operation is performed between the contents of the A register and the contents of the memory location as addressed by the operand address in location n + 1. The result is placed in the A register. If the same bit position of the memory location and the A register contains a zero, or if both bit positions contain a one, the result is zero. The truth table is shown below, where n = bit position.

Timing: 3 cycles
Indexing: Yes
Indirect Addressing: Yes
Register Altered: A

Condition		Result
A (n)	Effective Memory Location (n)	A (n)
0	0	0
0	1	1
1	0	1
1	1	0

ANA **AND Memory and A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

15	M	A
----	---	---

The logical-AND is performed between the contents of the A register and the contents of the effective memory location. The result is placed in the A register. If the same bit position of both the effective memory location and A contain a one, the result is a one. The truth table is shown below, where n = bit position.

Timing: 2 cycles
Relative: Yes
Indexing: Yes
Indirect Addressing: Yes
Registers Altered: A

ANAI **AND to A Register Immediate**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	150
Operand		

A logical-AND is performed between the contents of the operand and the contents of the A register. The result is placed in the A register. If the same bit position of the operand and the A register contains a one, the result is one; otherwise, the result is zero. The truth table is shown below, where n = bit position.

Timing: 2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: A

ANAE **AND Memory and A Register Extended**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	6	15	X
I	Operand Address		

The logical-AND operation is performed between the contents of the A register and the contents of the memory location as addressed by the operand address in location n + 1. The result is placed in the A register. If the same bit position of both the memory location and the A register contains a one, the result is a one. The truth table is shown below, where n = bit position.

Timing: 3 cycles
Indexing: Yes
Indirect Addressing: Yes
Register Altered: A

Condition		Result
A (n)	Effective Memory Location (n)	A (n)
0	0	0
0	1	0
1	0	0
1	1	1

Jump Instructions

This group comprises the instructions that direct the program to a nonsequential address for execution of the instruction located there, but they neither mark the location (as do the jump-and-mark instructions) nor bring the program back to the main sequence (as do the execution instructions).

For the jump instruction group (except the special instructions IJMP, JSR, BT and SRE) the A field contains a set of nine flags which define the logical conditions for the execution of the jump function. The jump address is contained in the second word of the double-word instruction.

The jump condition is the logical-AND of all ones in the field. Thus, there are 512 possible combinations but not all are useful. The most useful conditional jump instructions are contained in the mnemonic instruction repertoire recognized by the DAS assembler.

jump instructions

JMP Jump Unconditionally

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				1				000							
I	Jump Address														

The next instruction executed is at the jump address.

Timing: 2 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JAP

Jump if A Register Positive

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1		002	
I	Jump Address				

If the contents of the A register are positive or zero, the next instruction executed is at the jump address. If the A register is negative, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JOF

Jump if Overflow Indicator Set

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				1				001							
I		Jump Address													

If the OF indicator is set, the next instruction executed is at the jump address. If the OF indicator is not set, the next instruction in sequence is executed. The OF indicator is reset upon execution of this instruction. If the condition is met.

Timing: Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: OF, P

JAN

Jump if A Register Negative

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				1				004					
I	Jump Address												

If the A register is negative, the next instruction executed is at the jump address. If the A register is positive or zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JAZ**Jump if A
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				1				010			
I		Jump Address									

If the A register is zero, the next instruction executed is at the jump address. If the A register is not zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
 Condition not met: 1 cycle
 Indexing: No
 Indirect Addressing: Yes
 Registers Altered: P

JBZ**Jump if B
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				1				020			
I		Jump Address									

If the B register is zero, the next instruction executed is at the jump address. If the B register is not zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
 Condition not met: 1 cycle
 Indexing: No
 Indirect Addressing: Yes
 Registers Altered: P

JXZ**Jump if X
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1		040	
I	Jump Address				

If the X register is zero, the next instruction executed is at the jump address. If the X register is not zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
 Condition not met: 1 cycle
 Indexing: No
 Indirect Addressing: Yes
 Registers Altered: P

JSS1**Jump if Sense
Switch 1 Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1		100	
I	Jump Address				

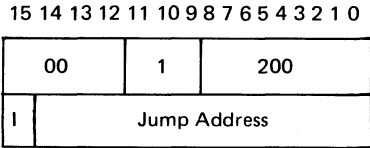
If sense switch 1 is set, the next instruction executed is at the jump address. If the sense switch being tested is not set, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
 Condition not met: 1 cycle
 Indexing: No
 Indirect Addressing: Yes
 Registers Altered: P

jump instructions

JSS2

Jump if Sense
Switch 2 Set

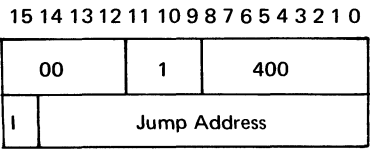


If sense switch 2 is set, the next instruction executed is at the jump address. If the sense switch being tested is not set, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle
Indexing: No
Indirect Addressing: Yes
Registers Altered: P

JSS3

Jump if Sense
Switch 3 Set

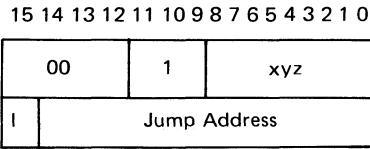


If sense switch 3 is set, the next instruction executed is at the jump address. If the sense switch being tested is not set, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle
Indexing: No
Indirect Addressing: Yes
Registers Altered: P

JIF

Jump if Combined
Conditions Are Met



If all the conditions specified by the xyz field are met, the next instruction executed is at the jump address. If they are not all met, the next instruction in sequence is executed. The values of x, y, and z are established by the first expression in the DAS variable field (see Section 21), and have the following meaning:

xyz (octal)	Jump Condition
0001	OFLO set
0002	$A \geq 0$
0004	$A < 0$
0010	$A = 0$
0020	$B = 0$
0040	$X = 0$
0100	SS1 set
0200	SS2 set
0400	SS3 set

Compound conditions may be specified by adding together the values of the desired conditions.

For example:

Instruction Field	Variable Field
JIF	0222, ALFA

Where 0222 = 0200 + 020 + 02 means: take the next instruction from address ALFA, if

jump instructions

and only if, all three of the following conditions are true:

The A register contains a positive number: 0002

The B register contains zero: 0020

Sense switch 2 is set: 0200

Timing: 2 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JBNZ

**Jump if B
Register not Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1		026	
I	Jump Address				

If the B register is not zero, the next instruction executed is at the jump address.
If the B register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycle

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JANZ

**Jump if A
Register not Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				1				016			
I		Jump Address									

If the A register is not zero, the next instruction executed is at the jump address.
If the A register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JXNZ

**Jump if X
Register not Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1	046	
I	Jump Address			

If the X register is not zero, the next instruction executed is at the jump address.
If the register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

jump instructions

JS1N

**Jump if Sense
Switch 1 not Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1	106
I	Jump Address		

If sense switch 1 is not set, the next instruction executed is at jump address. If the sense switch being tested is set, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JS3N

**Jump if Sense
Switch 3 not Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1	406
I	Jump Address		

If sense switch 3 is not set, the next instruction executed is at jump address. If the sense switch being tested is set, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

JS2N

**Jump if Sense
Switch 2 not Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		1	206
I	Jump Address		

If sense switch 2 is not set, the next instruction executed is at jump address. If the sense switch being tested is set, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

IJMP

Indexed Jump

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		6	70	X
I	Jump Address			

The indexed jump is an unconditional jump; therefore, the next executed instruction is at the indexed (post) jump address. Bit 2 is not used. The X field selects the register to be used for indexing as follows:

X	Address Mode
0 or 4	Relative to P
1 or 5	Indexed by X
2 or 6	Indexed by B
3 or 7	Direct or indirect (postindexing)

Timing: 2 cycles

Indexing: Yes — Post

Indirect Addressing: Yes

Registers Altered: P

**Jump Unconditionally
and Set Return in Index
Register**

JSR

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				C				50				X			
I		Jump Address													

The contents of the instruction P counter are stored in the specified index register. The next instruction executed is at the jump address. The index register to be used is specified by bit 1 of the X field. The remaining bits of the X field are not used.

<u>X</u>	<u>Index Register Used for Storage</u>
Bit 2 1 0	
* 1 * = 2,3,6 & 7	B
* 0 * = 0,1,4 & 5	X

Timing: 2 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: P (X, B)

JOFN

**Jump if Overflow
Indicator Not Set**

15 14 13 12 11 10 9 8 7 6 5 4 2 3 1 0

00				1		007			
I		Jump Address							

If the overflow indicator is not set, jumps to the instruction at the effective jump address and executes it next.

If the overflow indicator is set, executes the next instruction following JOFN.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect addressing: Yes

Registers altered: OF and P

BT *

Bit Test

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		6		4		Z		W	
I	Jump Address								

The bit test instruction will test the condition of a selected bit in the A or B register and jump if the condition is met. The bit to be tested is selected by the W field which can range from bit 0 to bit 15. The Z field has the following meaning:

<u>Z</u>	<u>Condition</u>
00	Bit in A = 1
01	Bit in B = 1
10	Bit in A = 0
11	Bit in B = 0

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

SRE *

Skip if Register Equal

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		6	6	Z	X
I	Operand or Operand Address				

The skip instruction will perform a logical compare between the register specified by the Z field (A, B, or X) and the memory word at the address specified by the second word. If the result is not equal, the next instruction in sequence is executed. If the result is equal, the next two locations are skipped and the instruction in the third location is executed. The Z fields for register selection are listed as follows:

<u>Z</u>	<u>Register Selected</u>
001	A
010	B
100	X

<u>X</u>	<u>Address Mode</u>
0 or 4	Relative to P
1 or 5	Indexed with X
2 or 6	Indexed with B
3 or 7	Direct or indirect (postindexing)

Timing:

No skip: 3 cycles

Skip: 3.5 cycles

Indexing: Yes — Post

Indirect Addressing: Yes

Registers Altered: None

***optional**

Jump-And-Mark Instructions

This group comprises the instructions that direct the program to a nonsequential jump address, store the contents of the P register there, and next execute the instruction following the jump address.

For the jump and mark group of instructions, the A field defines the same set of logical conditions specified for the jump group. Thus, there are 512 possible combinations, but not all are useful. The most useful instructions are contained in the mnemonic instruction repertoire recognized by the DAS assembler.

jump-and-mark instructions

JMPM

Jump and Mark Unconditionally

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	2	000
1	Jump Address	

The contents of the P counter are stored at the jump address. The next instruction executed is at the jump address plus one.

Timing: 3 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump address, P

NOTE: P counter is at the second word +1.

JOFM

Jump and Mark If Overflow Set

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2	001	
1	Jump Address			

If the OF indicator is set, the contents of the P counter are stored at the jump address, and the instruction at the jump address plus one is executed. If the OF indicator is not set, the next instruction in sequence is executed. The OF indicator is reset upon execution of the JOFM instruction.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P, OF

JANM

Jump and Mark if A Register Negative

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	2	004
1	Jump Address	

If the A register is negative, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the A register is positive or zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JAPM

Jump and Mark if A Register Positive

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2	002
1	Jump Address		

If the A register is positive or zero, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the A register is negative, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JAZM **Jump and Mark if A
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	2	010
I	Jump Address	

If the A register is zero, the contents of the P counter are placed at the jump address and the instruction at the jump address plus one is executed. If the A register is not zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JXZM **Jump and Mark if X
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	2	040
I	Jump Address	

If the X register is zero, the contents of the P counter are placed at the jump address and the instruction at the jump address plus one is executed. If the X register is not zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JBZM **Jump and Mark if B
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	2	020
I	Jump Address	

If the B register is zero, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the B register is not zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JS1M **Jump and Mark if
Sense Switch 1 Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	2	100
I	Jump Address	

If sense switch 1 is set, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the tested sense switch is not set, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

**Jump and Mark if
Sense Switch 2 Set**

JS2M

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2		200	
I	Jump Address				

If sense switch 2 is set, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the tested sense switch is not set, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

**Jump and Mark if
Sense Switch 3 Set**

JS3M

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2		400	
I	Jump Address				

If sense switch 3 is set, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the tested sense switch is not set, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

**Jump and Mark if Combined
Conditions Are Met**

JIFM

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2		xyz	
I	Jump Address				

If all of the conditions specified by the xyz field are met, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the conditions are not all met, the next instruction in sequence is executed. The values of x, y and z are established by the first expression in the DAS variable field (see Section 21). The jump conditions are the same as those specified for JIF, page 20-18.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JOFNM **Jump and Mark
if Overflow not Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2		007	
1	Jump Address				

If the OF indicator is not set, the contents of the P counter are stored at the jump address, and the instruction at the jump address plus one is executed. If the OF indicator is set, the next instruction in sequence is executed. The OF indicator is not reset upon execution of the JOFNM instruction.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JBNZM **Jump and Mark if B
Register not Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2		026	
1	Jump Address				

If the B register is not zero, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the B register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JANZM **Jump and Mark if A
Register not Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2	016	
1	Jump Address			

If the A register is not zero, the contents of the P counter are placed at the jump address and the instruction at the jump address plus one is executed. If the A register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

JXNZM **Jump and Mark if X
Register not Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2		046	
1	Jump Address				

If the X register is not zero, the contents of the P counter are placed at the jump address and the instruction at the jump address plus one is executed. If the X register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

jump and mark instructions

Jump and Mark if Sense Switch 1 not Set

JS1NM

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2	106
I	Jump Address		

If sense switch 1 is not set, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the tested sense switch is set, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

Jump and Mark if Sense Switch 2 not Set

JS2NM

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2	206
I	Jump Address		

If sense switch 2 is not set, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the tested sense switch is set, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

Jump and Mark if Sense Switch 3 not Set

JS3NM

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		2	406	
I	Jump Address			

If sense switch 3 is not set, the contents of the P counter are placed at the jump address, and the instruction at the jump address plus one is executed. If the tested sense switch is set, the next instruction in sequence is executed.

Timing:

Condition met: 3 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: Jump Address, P

Execution Instructions

This group comprises the instructions that direct the program to a nonsequential address for execution of the instruction located there, and then direct the program back to the main sequence to execute the instruction following the execution instruction.

These instructions contrast with the jump and jump-and-mark instructions in that the latter groups do not return the program to the main sequence but continue executing the instructions following the jump address.

For the execute group of instructions, the A field contains a set of nine flags which define the logical conditions for executing an instruction contained at the effective execution address. The execution address is contained in the second word of the double-word instruction. The execute condition is the logical-AND of all ones in the A field. The most useful of the 512 possible execute instructions are contained in the mnemonic instruction repertoire recognized by the DAS assembler.

It is important to note that only single-word instructions which do not specify relative addressing should be executed. The single-word instruction groups are load/store, arithmetic, logical, control, shift, and register change.

If the execution is attempted on double-word instructions, erroneous operations will occur.

execution instructions

XEC Execute Unconditionally

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		000	
I	Execute Address				

The instruction located at the execute address is executed and then the next instruction in sequence is executed.

Timing: 2 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XAP Execute if A Register Positive

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		002	
I	Execute Address				

If the A register is positive or zero, the instruction at execute address is executed, and then the next instruction in sequence is executed. If the A register is negative, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XOF Execute if Overflow Set

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		001	
I	Execute Address				

If the OF indicator is set, the instruction at the execute address is executed, and then the next instruction in sequence is executed.

If the OF indicator is not set, the next instruction in sequence is executed. Execution of the XOF instruction resets the OF indicator.

Timing: Condition met: 2 cycles

Condition not met: 1 cycle

Timing: 2 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: OF (reset)

XAN Execute if A Register Negative

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		004	
I	Execute Address				

If the A register is negative, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the A register is positive, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles

Condition not met: 1 cycle

Timing: 2 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XAZ **Execute if A
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				3				010							
I	Execute Address														

If the A register is zero, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the A register is not zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle
Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XXZ **Execute if X
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		040	
I	Execute Address				

If the X register is zero, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the index register is not zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle
Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XBZ **Execute if B
Register Zero**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		020	
I	Execute Address				

If the B register is zero, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the B register is not zero, the next instruction in sequence is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle
Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XS1 **Execute if Sense
Switch 1 Set**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		100	
I	Execute Address				

If sense switch 1 is set, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the sense switch tested is not set, the next instruction is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle
Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XS2 **Execute if Sense
Switch 2 Set**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00				3				200							
I	Execute Address														

If sense switch 2 is set, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the sense switch tested is not set, the next instruction is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle

Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XS3 **Execute if Sense
Switch 3 Set**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00				3				400							
I	Execute Address														

If sense switch 3 is set, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the sense switch tested is not set, the next instruction is executed.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle

Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XIF **Execute if Combined
Conditions Are Met**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00				3				xyz							
I	Execute Address														

If all of the conditions specified by the xyz field are met, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the conditions are not all met, the next instruction is executed. The values of x, y, and z are established by the first expression in the DAS variable field (see Section 21). The execute conditions are the same as jump conditions specified for JIF, page 20-18.

Timing: Condition met: 2 cycles
Condition not met: 1 cycle

Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XOFN **Execute if Overflow
not Set**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00				3				007							
I	Execute Address														

If the OF indicator is not set, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the OF indicator is set, the next instruction in sequence is executed. Execution of the XOFN instruction does not reset the OF indicator.

Timing:
Condition met: 2 cycles
Condition not met: 1 cycle

Indexing: No
Indirect Addressing: Yes
Registers Altered: None

XANZ **Execute if A**
Register not Zero

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		016	
I	Execute Address				

If the A register is not zero, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the A register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XXNZ **Execute if X**
Register not Zero

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00				3				046							
I		Execute Address													

If the X register is not zero, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the X register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XBNZ **Execute if B**
Register not Zero

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3		026	
I	Execute Address				

If the B register is not zero, the instruction at the execute address is executed, and then the next instruction in sequence is executed. If the B register is zero, the next instruction in sequence is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XS1N **Execute if Sense**
Switch 1 not Set

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00		3	106
I	Execute Address		

If sense switch 1 is not set, the instruction at the execute address is executed and then the next instruction in sequence is executed. If the sense switch tested is set, the next instruction is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

execution instructions

XS2N Execute if Sense Switch 2 not Set

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00				3				206							
I	Execute Address														

If sense switch 2 is not set, the instruction at the execute address is executed and then the next instruction in sequence is executed. If the sense switch tested is set, the next instruction is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

XS3N Execute if Sense Switch 3 not Set

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00				3				406							
I	Execute Address														

If sense switch 3 is not set, the instruction at the execute address is executed and then the next instruction in sequence is executed. If the sense switch tested is set, the next instruction is executed.

Timing:

Condition met: 2 cycles

Condition not met: 1 cycle

Indexing: No

Indirect Addressing: Yes

Registers Altered: None

Control Instructions

This group consists of general control instructions for the CPU.

All four instructions follow the single-word, non-addressing format (Section 18).

control instructions

HLT

Halt

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	0	XXX
----	---	-----

When the computer executes the halt instruction, computation is stopped and the computer is placed in the step mode. When the RUN button is pressed, computation starts with the next instruction in sequence.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: None

NOP

No Operation

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	000
----	---	-----

Execution of the NOP instruction does not affect the A, B, and X registers or memory.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: None

SOF

Set Overflow Indicator

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	7	401
----	---	-----

The OF indicator is set.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: OF

ROF

Reset Overflow Indicator

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	7	400
----	---	-----

The OF indicator is reset.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: OF

Shift Instructions

Shift instructions all follow the single-word, non-addressing format (Section 18).

For shift instructions, the address field A defines the type of shift (bits 5-8) and the number of bit positions to be shifted (bits 0-4). Twelve of the possible 16 shift operations defined by bits 5-8 are implemented.

Timing for all instructions in the shift instruction group can be determined as follows:

$$\begin{aligned}\text{Timing} &= 1 \text{ cycle, for } n = 0 \text{ or } 1 \\ &= 1 + 0.147 (n-1) \text{ cycles,} \\ &\quad \text{for } n = 02 \text{ through } 037\end{aligned}$$

where n equals the number of shifts.

shift instructions

Logical Shift Right A Register

LSRA

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$340 + n$
----	---	-----------

The contents of the A register are shifted n places to the right ($n = 0$ to 037). Zeros are shifted into the high-order positions of the A register. Information shifted out of the low-order positions of the A register is lost.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A

Logical Rotate Left A Register

LRLA

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$240 + n$
----	---	-----------

The contents of the A register are rotated n places to the left ($n = 0$ to 037). Bit position 15 of the A register is rotated into bit position 0.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A

Logical Shift Right B Register

LSRB

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$140 + n$
----	---	-----------

The contents of the B register are shifted n places to the right ($n = 0$ to 037). Information shifted out of the low-order position of the B register is lost. Zeros are shifted into the high-order position of the B register.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: B

Logical Rotate Left B Register

LRLB

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$040 + n$
----	---	-----------

The contents of the B register are rotated n positions to the left ($n = 0$ to 037). Bit position 15 of the B register is rotated into bit position 0.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: B

*See page 20-29.

LLSR

**Long Logical
Shift Right**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$540 + n$
----	---	-----------

The contents of the A and B registers are shifted right n positions ($n = 0$ to 037). Bits shifted out of the low-order position of B are lost. Zeros are shifted into the high-order position of the A register.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A, B

ASRA

**Arithmetic Shift
Right A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$300 + n$
----	---	-----------

The contents of the A register are shifted n positions to the right ($n = 0$ to 037). Bits shifted out of the low-order positions of A are lost. The sign bit (bit 15) of the A register is extended n places to the right.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A

LLRL

**Long Logical
Rotate Left**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$440 + n$
----	---	-----------

The contents of the A and B registers are rotated n positions to the left ($n = 0$ to 037). Bit position 15 of the A register is rotated into bit position 0 of the B register.

Timing*

Indexing: No

Indirect Address: No

Registers Altered: A, B

ASLA

**Arithmetic Shift
Left A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	$200 + n$
----	---	-----------

The contents of the A register are shifted n places to the left ($n = 0$ to 037). The sign bit (bit 15) of the A register is retained and zeros are shifted into the low-order positions of the A register. Bits shifted out of bit position 14 are lost.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A

*See page 20-29.

LASR **Long Arithmetic Shift Right**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	500 + n
----	---	---------

The contents of the A and B registers are shifted n places to the right ($n = 0$ to 037). Bit position 0 of the A register is shifted into bit position 14. The sign bit of the A register (bit 15) is extended n places to the right. The sign bit (bit 15) of the B register remains unchanged.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A, B

ASRB **Arithmetic Shift Right B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	100 + n
----	---	---------

The contents of the B register are shifted n places to the right ($n = 0$ to 037). Information shifted out of the low-order position of B is lost. The sign bit (bit 15) of the B register is extended n places to the right.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: B

LASL **Long Arithmetic Shift Left**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	400 + n
----	---	---------

The contents of the A and B registers are shifted n places to the left ($n = 0$ to 037). Bit position 14 of the B register is shifted into bit position 0 of the A register; the sign bit (bit 15) of the B register remains unchanged. The sign bit (bit 15) of the A register also remains unchanged. The bit shifted out of bit position 14 of the A register is lost, and zeros are shifted into the low-order positions of the B register.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: A, B

ASLB **Arithmetic Shift Left B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	4	000 + n
----	---	---------

The contents of the B register are shifted n places to the left ($n = 0$ to 037). The sign bit (bit 15) of the B register is retained and zeros are shifted into the low-order positions of the B register. Bits shifted out of bit position 14 are lost.

Timing*

Indexing: No

Indirect Addressing: No

Registers Altered: B

*See page 20-29

Register-Change Instructions

The facility to operate on and copy data between the A, B, and X registers is available in the general register change instruction group.

All instructions in this group use the same op code and M field code (00 and 5, respectively). Bits 0 to 8 of the A field specify the register and the function.

The programmer can specify combinations of address bits to perform simultaneous operations. If the selected address bits specify that more than one register is the source, these are logically ORed. The function of each address bit is shown below.

Indirect addressing and indexing do not apply to this instruction group. Each instruction requires one machine cycle regardless of the number of functions performed. As an aid to the programmer, the most useful bit combinations have mnemonics assigned to them.

Bit	Function
0	A is a destination register
1	B is a destination register
2	X is a destination register
3	A is a source register
4	B is a source register
5	X is a source register
8	Conditions the function with the OF indicator. If bit 8 is one, the function is performed only if OF is set. If bit 8 is zero, the function is performed unconditionally. The OF register does not change.

Bits 6 and 7 indicate the function to be performed as follows:

Bit	Bit	Function
7	6	
0	0	Transfer
0	1	Transfer incremented
1	0	Transfer complemented
1	1	Transfer decremented

register-change instructions

IAR	Increment A Register
-----	-------------------------

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	111
----	---	-----

DAR **Decrement A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	311
----	---	-----

IBR	Increment B Register
00000000	00000000
00000001	00000001
00000010	00000010
00000011	00000011
00000100	00000100
00000101	00000101
00000110	00000110
00000111	00000111
00001000	00001000
00001001	00001001
00001010	00001010
00001011	00001011
00001100	00001100
00001101	00001101
00001110	00001110
00001111	00001111
00010000	00010000
00010001	00010001
00010010	00010010
00010011	00010011
00010100	00010100
00010101	00010101
00010110	00010110
00010111	00010111
00011000	00011000
00011001	00011001
00011010	00011010
00011011	00011011
00011100	00011100
00011101	00011101
00011110	00011110
00011111	00011111
00100000	00100000
00100001	00100001
00100010	00100010
00100011	00100011
00100100	00100100
00100101	00100101
00100110	00100110
00100111	00100111
00101000	00101000
00101001	00101001
00101010	00101010
00101011	00101011
00101100	00101100
00101101	00101101
00101110	00101110
00101111	00101111
00110000	00110000
00110001	00110001
00110010	00110010
00110011	00110011
00110100	00110100
00110101	00110101
00110110	00110110
00110111	00110111
00111000	00111000
00111001	00111001
00111010	00111010
00111011	00111011
00111100	00111100
00111101	00111101
00111110	00111110
00111111	00111111
01000000	01000000
01000001	01000001
01000010	01000010
01000011	01000011
01000100	01000100
01000101	01000101
01000110	01000110
01000111	01000111
01001000	01001000
01001001	01001001
01001010	01001010
01001011	01001011
01001100	01001100
01001101	01001101
01001110	01001110
01001111	01001111
01010000	01010000
01010001	01010001
01010010	01010010
01010011	01010011
01010100	01010100
01010101	01010101
01010110	01010110
01010111	01010111
01011000	01011000
01011001	01011001
01011010	01011010
01011011	01011011
01011100	01011100
01011101	01011101
01011110	01011110
01011111	01011111
01100000	01100000
01100001	01100001
01100010	01100010
01100011	01100011
01100100	01100100
01100101	01100101
01100110	01100110
01100111	01100111
01101000	01101000
01101001	01101001
01101010	01101010
01101011	01101011
01101100	01101100
01101101	01101101
01101110	01101110
01101111	01101111
01110000	01110000
01110001	01110001
01110010	01110010
01110011	01110011

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	122
----	---	-----

DBR

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	322
----	---	-----

IXR	Increment X Register
-----	----------------------

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	144
----	---	-----

The contents of the A (B, X) register are incremented by one. When the value of the A (B, X) register is 077777 (maximum positive number) and IAR (IBR, IXR) is executed, OF is set and the value of A (B, X) is 100000.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A (B, X), OF

DXR	Decrement X Register
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	344
----	---	-----

The contents of the A (B, X) register are decremented by one. When the value of the A (B, X) register is 100000 and DAR (DBR, DXR) is executed, OF is set and A (B, X) is 077777.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A (B, X), OF

CPA

**Complement A
Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	211
----	---	-----

CPB

**Complement B
Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	222
----	---	-----

CPX

**Complement X
Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	244
----	---	-----

The contents of the A (B, X) register are complemented (one's complement).

Timing: 1 cycle
Indexing: No
Indirect Addressing: No
Registers Altered: A (B, X)

TAB

**Transfer A
Register to
B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	012
----	---	-----

The contents of the A register are placed in the B register.

Timing: 1 cycle
Indexing: No
Indirect Addressing: No
Registers Altered: B

TAX

**Transfer A
Register to
X Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	014
----	---	-----

The contents of the A register are placed in the X register.

Timing: 1 cycle
Indexing: No
Indirect Addressing: No
Registers Altered: X

TBA

**Transfer B
Register to
A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	021
----	---	-----

The contents of the B register are placed in the A register.

Timing: 1 cycle
Indexing: No
Indirect Addressing: No
Registers Altered: A

TBX

**Transfer B
Register to
X Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	024
----	---	-----

The contents of the B register are placed in the X register.

Timing: 1 cycle
Indexing: No
Indirect Addressing: No
Registers Altered: X

register-change instructions

TXA

**Transfer X
Register to
A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	041
----	---	-----

The contents of the X register are placed in the A register.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A

TXB

**Transfer X
Register to
B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	042
----	---	-----

The contents of the X register are placed in the B register.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: B

TZA

**Transfer Zero
to A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	001
----	---	-----

TZB

**Transfer Zero
to B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	002
----	---	-----

TZX

**Transfer Zero
to X Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	004
----	---	-----

The A (B, X) register is cleared to zero.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A (B, X)

AOFA

**Add Overflow
to A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	511
----	---	-----

AOFB

**Add Overflow
to B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	522
----	---	-----

AOFX

**Add Overflow
to X Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	544
----	---	-----

The content of the OF indicator is added to the A (B, X) register. The sum is placed in the A (B, X) register. The overflow flip-flop does not change.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A (B, X)

SOFA

**Subtract Overflow
from A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	711
----	---	-----

SOFB

**Subtract Overflow
from B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	722
----	---	-----

SOFX

**Subtract Overflow
from X Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	744
----	---	-----

The content of the OF indicator is subtracted from the A (B, X) register. The overflow flip-flop does not change.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A (B, X)

MERGE

Merge Source
to Destination

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	xyz
----	---	-----

The inclusive-OR of the contents of all the specified source registers is placed in each of the specified destination registers. The values of x, y, and z are established by the DAS variable field (Section 21). The meaning for each bit in the field is given in the chart on page 20-33.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A, B, X, as specified

DECR

Decrement Source
to Destination

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	xyz
----	---	-----

+1 is subtracted from the inclusive-OR of the contents of all the specified source registers and the result is placed in each of the specified destination registers. The values of x, y, and z are established by the DAS variable field (Section 21). The meaning for each bit in the field is given in the chart on page 20-33.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A, B, X, as specified

INCR

Increment Source
to Destination

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	xyz
----	---	-----

+1 is added to the inclusive-OR of the contents of all the specified source registers and the result is placed in each of the specified destination registers. The values of x, y, and z are established by the DAS variable field (Section 21). The meaning for each bit in the field is given in the chart on page 20-33.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A, B, X, as specified

COMPL

Complement Source
to Destination

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	xyz
----	---	-----

The inclusive-OR of the contents of all the specified source registers is ones-complemented and the result is placed in each of the specified destination registers. The values of x, y, and z are established by the DAS variable field (Section 21). The meaning for each bit in the field is given in the chart on page 20-33.

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A, B, X, as specified

ZERO**Zero Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

00	5	00z
----	---	-----

Each of the destination registers specified by **z** (page 20-33) is set to zero. The value **z** is established by the DAS variable field (Section 21).

Timing: 1 cycle

Indexing: No

Indirect Addressing: No

Registers Altered: A, B, X, as specified

I/O Instructions

The I/O instruction group is used for all communication between the computer and the peripheral devices that supply and receive data. Their use is described in detail in Sections 11 and 12.

There are three basic types of I/O instructions, as presented in the following pages:

- Control (including the Sense and External instructions).
- Data transfer between the external device and the CPU registers.
- Data transfer between the external device and memory.

Each I/O instruction must specify the address of the peripheral device that is affected. Standard device addresses for the most commonly used I/O devices are given on page 20-44.

SEN**Program Sense**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10		1	XYY	
I	Jump Address			

I = 0, word contains an address

I = 1, word contains an indirect address

The sense instruction senses the logical state of an external line.

The nine bits represented by XXX are placed on the party-line I/O bus and represent the condition to be tested. X defines a specific line within device YY. The associated peripheral controller replies with a true or false signal.

If a true signal is received by the 620/f, a jump is made to the jump address. If a false signal is received, the next instruction in sequence is executed.

Timing: No response: 2.4 to 3.8 cycles
With response: 3.2 to 4.6 cycles

Indexing: No

Indirect Addressing: Yes

Registers Altered: P

EXC**External Control**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10				0	XXX									
----	--	--	--	---	-----	--	--	--	--	--	--	--	--	--

The external control instruction places a function code, contained in bits 0–8, on the E bus to initiate a control operation in an external device.

The nine bits represented by XXX are placed on the E bus for transmission to the peripheral controllers. The device address is contained in the YY portion of the data, and the function to be performed by the selected device is contained in the X portion.

Timing: 2.4 to 3.8 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: None

CIA**Clear and Input to
A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10				2	5ZZ									
----	--	--	--	---	-----	--	--	--	--	--	--	--	--	--

The A register is cleared and a data word from the selected device, ZZ, is transferred to the A register.

Timing: 4.9 to 6.2 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: A

CIB**Clear and Input to
B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	6ZZ
----	---	-----

The B register is cleared and a data word from the selected device, ZZ, is transferred to the B register.

Timing: 4.9 to 6.2 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: B

INA**Input to A Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	1ZZ
----	---	-----

A data word from the selected device, ZZ, is inclusively ORed with the contents of the A register.

Timing: 4.9 to 6.2 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: A

CIAB**Clear and Input to
A and B Registers**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	7ZZ
----	---	-----

The A and B registers are cleared and a data word from the selected device, ZZ, is transferred to the A and B registers.

Timing: 4.9 to 6.2 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: A and B

INB**Input to B Register**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	2ZZ
----	---	-----

A data word from the selected device, ZZ, is inclusively ORed with the contents of the B register.

Timing: 4.9 to 6.2 cycles

Indexing: No

Indirect Addressing: No

Registers Altered: B

INAB Input to A and B Registers

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	3ZZ
----	---	-----

A data word from the selected device, ZZ, is inclusively ORed with the inclusive – OR of the contents of the A and B registers; the results are transferred into both the A and B registers, i.e., $(A) + (B) + ZZ \rightarrow A, B$

Timing: 4.9 to 6.2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: A and B

OAR Output from A Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	3	1ZZ
----	---	-----

The contents of the A register are transferred to the selected device, ZZ.

Timing: 4.9 to 6.2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: None

OBR Output from B Register

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	3	2ZZ
----	---	-----

The contents of the B register are transferred to the selected device, ZZ.

Timing: 4.9 to 6.2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: None

OAB Output from A and B Registers

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	3	3ZZ
----	---	-----

The inclusive OR of the contents of the A and B registers is transferred to the selected device, ZZ.

Timing: 4.9 to 6.2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: None

IME	Input to Memory	OME	Output from Memory
------------	------------------------	------------	---------------------------

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	2	0ZZ
I	Data Address	

I = 0, since indirect addressing
is not permitted

A data word from the selected device, ZZ, is placed in the cleared effective memory address.

Timing: 5.9 to 7.2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: Memory

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

10	3	0ZZ
I	Data Address	

I = 0, since indirect addressing
is not permitted.

The contents of the effective memory location are transferred to the selected device, ZZ.

Timing: 5.9 to 7.2 cycles
Indexing: No
Indirect Addressing: No
Registers Altered: None

Class Codes	Addresses	Option or Controller
00-07	01-07	Teletype controller (620-06, 07, -08)
10-17	10-13	Magnetic tape controller (620-30, -31)
	14	Fixed-head rotating memory (620-38, -42 through -49)
	15	Movable-head rotating memory (620-40, -41)
	16, 17	Movable-head rotating memory (620-39, -37)
20-27	20, 21	First buffer interlace controller (620-20)
	22, 23	Second buffer interlace controller
	24, 25	Third buffer interlace controller
	26, 27	Fourth buffer interlace controller
30-37	30	Card reader (620-22 and 620-25)
	31	Card punch (620-26/27)
	32	Digital plotter (620-72)
	33	Electrostatic plotter
	34	Second paper tape system
	35, 36	Line printer (620-77)
	37	First paper tape system (620-51, -53, -55)
40-47	40-44, 46	Priority interrupt module (620-16)
	45	Memory protection (620-05)
	47	Real-time clock (620-13)
50-57	50-53	Special applications
	54-57	Analog system (620-85A, -85O, -85I)
60-67	60-67	Digital I/O (620-81), buffered I/O (620-80)
70-77	70-73	Communications controller (620-60, -61, -65, -66)
	74-77	Relay I/O (620-83)

Figure 20-1. Standard Device Addresses (Octal)

SECTION 21 – VARIAN DAS ASSEMBLERS

The Varian 620/f assembler language (DAS) specifies instructions, addresses, address modifications, and constants in a straightforward and meaningful manner. Instruction mnemonics replace the usual numerical codes. Memory addresses can be referenced symbolically, thus offering a flexibility not attainable with absolute addressing. Constants can be used without conversion to binary or octal values. For ease in checkout and documentation, comments can be added between symbolic statements, or appended to the statements themselves.

DAS coding reduces machine language bookkeeping without compromising full utilization of the computer.

DAS 4A operates in a Varian 620/f minimum-configuration system comprising the computer, 4K memory, and an on-line Teletype (TTY). DAS 8A requires 8K memory. Both can use and enhance additional system components, e.g., magnetic or paper tape equipment, card equipment, additional memory, line printer, etc. DAS MR operates under control of the Varian Master Operating System (MOS).

DAS translates symbolically coded instructions and data (source program) into binary instructions and data (object program).

The DAS Character Set

The DAS character set comprises:

a. Alphabetical characters

ABCDEFGHIJKLMNOPQRSTUVWXYZ
VWXYZ\$

b. Numerical characters

0123456789

c. TTY characters

CR (carriage return)
LF (line feed)

d. Special characters

+ (plus sign)
- (minus sign)
* (asterisk)
/ (slash)
. (period)
(blank)
@ (at sign)
[(left bracket)
] (right bracket)
< (less-than)
> (greater-than)
↑ (up arrow)
← (left arrow)
? (question mark)
= (equal sign)
, (comma)
' (prime)
((left parenthesis)
) (right parenthesis)
\ (back slash)
! (exclamation point)
" (quotation mark)
(pound sign)

DAS statement format

%	(percent sign)
&	(ampersand)
:	(colon)
;	(semicolon)

Statement Format

Each DAS source statement comprises a combination of label, operation, variable, and comment fields depending on the requirements of the instruction or directive.

Label Field

Symbols in the label field comprise one to four alphanumeric characters for the DAS 4A and DAS 8A assemblers and from one to six such characters for the DAS MR assembler. In any case, the first character is alphabetical. While only the given number of characters are recognized by the assembler, additional characters can be added.

Symbols are usually attached only to those source statements referenced elsewhere in the program, but this is not mandatory.

Operation Field

This field contains mnemonics for computer instructions and assembler directives. An asterisk following the mnemonic indicates indirect addressing. The mnemonics can be redefined with OPSY directives.

Variable Field

The purpose of this field varies according to the requirements of the operation. The variable field can consist of a symbol, a constant, or an expression combining symbols and constants. Expressions in DAS are similar to arithmetic expressions except that parentheses do not appear. The variable field can contain the following operators: + (addition), - (subtraction), * (multiplication), and / (division).

Arithmetic operations always involve all 16 bits of the words. Operations are performed from left to right, with multiplication and division being performed before addition and subtraction. Thus, $A + B/C * D$ in DAS is equivalent to $A + (B/C) * D$ in conventional notation.

Use of the asterisk in the variable field gives access to the current value of the location counter. This asterisk is translated as the current value of the location counter, i.e., * + 1 means the current value of the location counter plus one.

DAS constants are described in the following sections, wherein unsigned numbers are considered positive. DAS recognizes decimal and octal integers; floating-point numbers; alpha, address, and indirect constants; and literals.

A decimal integer is a signed or unsigned string of decimal digits, the first of which is not zero.

Examples:

1	29	-3	-9000
---	----	----	-------

An octal integer is a signed or unsigned string of octal digits, the first of which is zero.

Examples:

07	-044	+022745
----	------	---------

A floating-point number has the form:

$\pm \text{integer.fraction} E^{\pm \text{exponent}}$

where the right parenthesis, at least one digit, and the decimal point are always present. Other items in the format are optional.

Examples:

)375.64E + 7	)9.E -2, .1E + 12
) -4. + 20	

An **alpha constant** is a string of characters within primes (''). Internally, it is represented in eight-bit ASCII code. Each memory word can contain two characters, where blanks are also recognized as characters. In the DAS 8A assembler, an alpha constant can be a term in an arithmetic expression. However, if more than one word is generated by the constant, only the last word generated is subject to arithmetic manipulation.

Examples:

'A'*04000	'AB' + 1	'ABCD' + 011
-----------	----------	--------------

where, in the last example, two words are generated and 011 added to the second word.

An **address constant** is a symbol, number, or expression enclosed in parentheses. It generates a 15-bit direct address (bit 15 = 0).

Examples:

(a + 2)	(3)	(a)
---------	-----	-----

where a is an address symbol whose value is taken from the symbol table.

An **indirect address constant** is an address constant within parenthesis and followed by an asterisk. It generates a 15-bit indirect address (bit 15 = 1).

Examples:

(a + 2)*	(3)*	(a)*
----------	------	------

where a is an address symbol whose value is taken from the symbol table.

A **literal** comprises an expression preceded by an equal sign.

Examples:

=3	=+3	=-044	=(a + 2)*
= 'A'	= 'GO'		

Literals permit reference to a constant in the variable field where the assembler generates the data and assigns memory addresses. Even where a literal is used many times, only one memory address is assigned for each literal. Note that the expressions reflect the values assigned to the symbols rather than the contents of the memory addresses referenced by the symbols.

Comments Field

The comments field is separated from the variable field by at least one blank. It is used for any desired comments and is ignored by the assemblers.

Modes of Symbols and Expressions

Each symbol or expression has one of the modes external (E), common (C), relative (R), or absolute (A), assigned by the assembler. The mode of an expression is determined by the mode of the symbols in the expression.

The mode of a symbol is determined by the following rules:

- If the symbol is defined by an EXT directive, the mode is E.
- If the symbol is defined by a COMN directive, the mode is C.
- If the symbol is a symbol in a program, or if * is the current location counter value, the mode is R.

DAS instructions

- d. If the symbol is a numerical constant, the mode is A.
- e. If the symbol is defined by EQU, SET, or similar directive, the mode of the symbol is that of the variable field expression in the directive.

The mode of an expression is determined by the following rules:

- a. If the expression contains any mode E or C symbol, the expression is mode E.
- b. If the expression contains only mode A symbols, the expression is mode A.
- c. If the expression contains mode A and R symbols, the mode of the expression is R if there is an odd number of mode R symbols. Otherwise, the mode of the expression is A.

The following restrictions apply only to DAS MR and to FORTRAN-compatible output assembly within DAS 8A.

- a. No expression can contain symbols of both modes E and C.
- b. A mode E expression comprises a single mode E symbol.
- c. No mode E, C, or R expression can multiply or divide a mode E or C symbol.
- d. No expression can add or subtract a mode C and a mode R symbol, or a mode E and a mode R symbol.
- e. No expression can add two or more mode E, C, or R symbols.
- f. A mode A symbol can be added to or subtracted from a mode C or R symbol.

Figure 21-1 illustrates the above rules.

Assembler Instructions

DAS assemblers recognize all Varian 620 instructions, even when the system lacks the

EEEE	EXT		EEEE defined as type E
CCCC	COMN	6	CCCC defined as type C
RTN	ENTR		RTN is type R, a symbol
TBL	BSS	50	TBL is type R
ABL	BSS	'A' + 5	ABL is type R
LENG	EQU	*-TBL	LENG is type A, length of area
		CALL	EEEE, TBL, LENG
	LDA	* + 6	OK, relative forward
	LDA	CCCC + 6	Illegal, one word inst, not R or A
	LDXI	CCCC + 6	OK, two word instruction
	LDA	0, 1	Get CCCC + 6 to A, legal
	DATA	EEEE + 4	Illegal, value not zero
	DATA	CCCC + 4	Legal
	DATA	CCCC + LENG	Legal
	DATA	TBL + LENG -5	Legal, mode is R

Figure 21-1. Manipulation of Expression and Symbol Modes

hardware for execution. The programmer is responsible for knowing the instructions applicable to his system.

All DAS assembler instructions have the format:

- label field
- operation field
- variable field

where the label field is optional and contains a symbol when used, the operation field contains the instruction mnemonic, and the variable field contains one, two, or three expressions. Where there are two or three expressions, they are separated by commas.

Assembler instructions are of five types described below and in Figure 21-2. Figure 21-3 lists the instruction mnemonics by type.

Addressing of Single Word

If an address lower than 2048 is specified without indirect addressing, the assembler

generates an instruction with direct addressing.

If indexing is specified, the assembler generates an indexed instruction.

If the address specified is 1 to 512 words (inclusive) beyond the current instruction, the assembler generates an instruction with relative addressing.

If indirect addressing with a data address lower than 512 is specified, the assembler generates an instruction with indirect addressing and the specified address.

In all other cases, including indirect addressing with an address higher than 511, the assembler generates an instruction with indirect addressing and the specified address, stores the address in a table, and inserts the storage address in the referencing instruction. Duplicate values in the table are discarded.

Instruction Types

Figure 21-2 gives the characteristics of the

	Type 1	Type 2	Type 3	Type 4	Type 5
Words generated	1	2	2	1	2
Expressions in the Variable field	1 to 2	1	2	1	1 to 3
Memory addressing	Yes	Yes	Yes	No	Yes
Indirect addressing	Yes	Yes	Yes	No	Yes
Indexing	Yes	No	No	No	Yes
Condition micro-coding	No	No	Yes	Yes	No

Figure 21-2. Characteristics of Assembler Instruction Types

Type 1	Type 2	Type 3	Type 4	Type 5		
ADD	ADDI	JS3N	BT	AOFA	LRLB	ADDE
ANA	ANAI	JS3NM	IME	AOFB	LSRA	ANAE
DIV	DIVI	JXNZ	JIF	AOFX	LSRB	DIVE
ERA	ERAI	JXNZM	JIFM	ASLA	MERG	ERAE
INR	INRI	JXZ	JMIF	ASLB	NOP	IJMP
LDA	JAN	JXZM	OME	ASRA	OAB	INRE
LDB	JANM	LDAI	SEN	ASRB	OAR	JSR
LDX	JANZ	LDBI	XIF	CIA	OBR	LDAE
MUL	JANZM	LDXI		CIAB	ROF	LDBE
ORA	JAP	MULI		CIB	SEL	LDXE
STA	JAPM	ORAI		COMP	SEL2	MULE
STB	JAZ	STAI		CPA	SOF	ORAE
STX	JAZM	STBI		CPB	SOFA	SRE
SUB	JBNZ	STXI		CPX	SOFB	STAE
	JBNZM	SUBI		DAR	SOFX	STBE
	JBZ	XAN		DBR	TAB	STXE
	JBZM	XANZ		DECR	TAX	SUBE
	JMP	XAP		DXR	TBA	
	JMPM	XAZ		EXC	TBX	
	JOE	XBNZ		EXC2	TSA	
	JOEM	XBZ		HLT	TXA	
	JOEN	XEC		IAR	TXB	
	JOENM	XOF		IBR	TZA	
	JSS1	XOFN		INA	TZB	
	JSS2	XS1		INAB	TZX	
	JSS3	XS1N		INB	ZERO	
	JS1M	XS2		INCR		
	JS1N	XS2N		IXR		
	JS1NM	XS3		LASL		
	JS2M	XS3N		LASR		
	JS2N	XXNZ		LLRL		
	JS2NM	XXZ		LLSR		
	JS3M			LRLA		

Figure 21-3. Assembler Instructions by Type

types of assembler instructions and Figure 21-3 lists the instructions by type.

A **type 1 instruction** occupies one computer word and is memory-addressing.

Indirect addressing is specified by an asterisk after the mnemonic or after a

variable field in parentheses.

Examples:

LDA *	expression
LDA	(expression) *

Indexing is specified by a variable field

having two expressions. The first is the indexing increment and is less than 512. The second specifies the indexing register where one is the X register and two is the B register.

Example:

LDA	0300,1
-----	--------

loads the A register with the contents of the memory address specified by the sum of the contents of the X register plus 0300. Thus, if the X register contains 0200, the operand for this instruction is in memory address 500 (0300 + 0200).

A type 2 instruction occupies two consecutive computer words and is memory-addressing. The second word is the address of a jump, jump-and-mark, or execution instruction; or the second word of an immediate instruction. The variable field contains only one expression.

Indirect addressing is specified as with an assembler type 1 instruction. There is no indexing with assembler type 2 instructions.

A type 3 instruction occupies two consecutive computer words and is memory-addressing. It differs from an assembler type 2 instruction in that the variable field contains two expressions.

For the JIF, JIFM, JMIF, and XIF instructions, the first expression specifies the condition(s) required for the jump, jump-and-mark, or execution. The conditions are specified according to the rules outlined in Section 20. Multiple conditions can be specified by setting additional bits.

Example:

JIF	0222,ALFA
-----	-----------

takes the next instruction from address ALFA if the A register contains a positive number (0002), the B register contains zero (0020), and SENSE switch 2 is set (0200) since $0222 = 0002 + 0020 + 0200$.

For the SEN, IME, and OME instructions, the first expression specifies the I/O device address and function.

For the BT instruction, the first expression specifies the register and bit to be tested.

Indirect addressing is specified by an asterisk after the mnemonic or after a variable field expression in parentheses.

Examples:

JIF*	0222,ALFA
JIF	0222,(ALFA)*

NOTE: Do not indirect address IME or ØME.

A type 4 instruction occupies one computer word and does not address memory.

For ZERO, DECR, MERG, COMP, INCR, and the register modification instructions using the format and coding given on page 20-33, the assembler generates an instruction as specified by the value in the variable field.

A type 5 instruction occupies two consecutive computer words and is memory-addressing.

Indirect addressing is specified by an asterisk after the mnemonic or after the first expression in the variable field when the expression is enclosed in parentheses.

Examples:

LDAE*	expression
LDAE	(expression)*
LDAE	(expression)*, expression

DAS directives

Pre-indexing is specified by two expressions in the variable field, where the first expression is the data address and the second specifies the indexing register: one for the X register, two for the B register.

Example:

LDAE	01300,2
------	---------

loads the A register with the contents of the memory address specified by the sum of the contents of the B register plus 01300. Thus, if the B register contains 0400, the operand for this instruction is in memory address 01700 (01300 + 0400).

Post-indexing is specified by three expressions in the variable field, where the first expression is the data address, the second specifies the indexing register as in pre-indexing, and the third is logically ORed with the instruction word.

Example:

LDAE	ADDR,2,0200
------	-------------

loads the A register extended and post-indexed with the B register. The assembler does not check the validity of the third expression, but its value cannot exceed 0200. Pre-indexing is not applicable to IJMP, JSR, or SRE.

Assembler Directives

Directives are instructions to the assembler. They are divided into the following groups:

- Symbol definition directives
- Instruction definition directives
- Location counter control directives
- Data definition directives

- Memory reservation directives
- Conditional assembly directives
- Assembler control directives
- Subroutine control directives
- List and punch control directives
- DAS 8A interface directive to stand-alone FORTRAN
- Program link directives
- MOS I/O control directives
- Macro definition directives

Directives have four fields:

- The label field, which contains a symbol. The label field is usually optional. However, some directives require it by the nature of their operations, e.g., EQU, SET, BEGI, and FORM.
- The operation field, which contains the directive mnemonic.
- The variable field, whose contents vary with the directive. Any symbols in this field that are not previously defined cause error flags to be output.
- The comment field (optional) for programming convenience.

In the following descriptions of the individual directives, the format

label field
operation field
variable field

is used, with the optional comment field

Directive	DAS 4A	DAS 8A	DAS MR
BEGI	Yes	Yes	No
BES	Yes	Yes	Yes
BSS	Yes	Yes	Yes
CALL	Yes	Yes	Yes
COMN	No	Yes	Yes
CONT	No	Yes	Yes
DATA	Yes	Yes	Yes
DETL	No	Yes	Yes
DUP	No	Yes	Yes
EJEC	No	Yes	Yes
EMAC	No	No	Yes
END	Yes	Yes	Yes
ENTR	Yes	Yes	Yes
EQU	Yes	Yes	Yes
EXT	No	Yes	Yes
FORM	No	Yes	Yes
FORT	No	Yes	No
GOTO	No	Yes	Yes
IFF	No	Yes	Yes
IFT	No	Yes	Yes
LIST	No	Yes	No
MAC	No	No	Yes
LOC	Yes	Yes	Yes
MAX	No	Yes	No
MIN	No	Yes	No
MORE	Yes	Yes	No
MZE	Yes	Yes	Yes
NAME	No	Yes	Yes
NLIS	No	Yes	No
NPUN	No	Yes	No
NULL	No	Yes	Yes
OPSY	No	Yes	Yes
ORG	Yes	Yes	Yes
PUNC	No	Yes	No
PZE	Yes	Yes	Yes
READ	No	Yes	No
RETU*	Yes	Yes	Yes

Figure 21-4. Directives Recognized by DAS Assemblers (1 of 2)

SET	Yes	Yes	Yes
SMRY	No	Yes	Yes
SPAC	No	Yes	Yes
USE	No	Yes	No

Figure 21-4. Directives Recognized by DAS Assemblers (2 of 2)

being understood to follow the variable field when used. In cases where the variable field contains more than one item or expression, these are always separated by commas:

Figure 21-4 shows the directives recognized by each DAS assembler, but does not include MOS I/O control directives accepted by the assembler.

Symbol Definition Directives

These directives assign arbitrary values to symbols in the symbol table. This table is a list of symbols in the source program. For each symbol in the table, there is a corresponding value, usually an address.

EQU (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol EQU expression

It places the symbol in the assembler's symbol table and assigns it the value of the expression. If the symbol has already been entered in the symbol table, error message DD is output and the value in the table is replaced by that of the expression. Any symbol used as the variable field expression must have been defined previously. The label field symbol is mandatory.

SET (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol SET expression

It is the same as EQU except that there is no error message output if the symbol has already been entered in the symbol table.

MAX (DAS 8A)

This directive has the format

symbol MAX expression(s)

It assigns to the symbol the largest algebraic value found among the expressions. Any symbol used as a variable field expression must have been defined previously. The label field symbol is mandatory. Use SET to redefine the symbol.

MIN (DAS 8A)

This directive has the format

symbol MIN expression(s)

It is the same as MAX except that the symbol is assigned the smallest algebraic value found among the expressions.

Instruction-Definition Directive

This directive redefines a standard instruction mnemonic.

OPSY (DAS 8A, DAS MR)

This directive has the format

symbol OPSY mnemonic

It makes the symbol a mnemonic with the mnemonic. The variable field must be a standard mnemonic.

Example:

CLA	OPSY	LDA
	CLA	BETA

Location Counter Control Directives

These directives control the location counter(s). DAS 8A has multiple location counters and directives to modify or preset their values. Figure 21-5 lists the five standard DAS 8A location counter symbols and their uses. They need not be created by the user. However, up to eight other loca-

tion counters in addition to those provided can be created, thus providing complex relocatable and overlay programs within a single assembly.

There are no user-created location counters at the beginning of an assembly. The assembler uses three location counters for location assignment. Thus, IAOR and LTOR are always in use, as is a third counter used to assign locations to generated instructions and generated data (except literals and indirect pointers). This is done by the blank location counter until USE specifies another.

In a straightforward program using only one location counter, complete control over the counter is maintained by ORG and LOC.

ORG (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol ORG expression

Counter	Initial Value	Use
SYOR	00000	Controls assignment of memory to all system parameters
IAOR	00200	Controls assignment of memory to indirect pointers
LTOR	01000	Controls assignment of memory to literals
COMN	02000	Controls assignment of memory within an interface area common to two or more programs
blank	04000	Used initially and normally by the assembler for memory assignments until/unless overridden by USE = 0

Figure 21-5. Standard DAS 8A Location Counters

DAS directives

where the symbol is optional. It sets the location counter currently in use to the value of the expression. If a symbol is present in the label field, it is set to the value of the expression. Any symbol used as the variable field expression must have been defined previously.

LOC (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol LOC expression

where the symbol is optional. It is used if the data and instructions following the LOC address are to be moved to the LOC address by the object program before execution, i.e., to keep a block of data or instructions undisturbed by assembly. LOC causes data or instructions following it to be generated as if there had been a ORG to change the current location counter value. However, this value is not actually changed. Any symbol used as a variable field expression must have been defined previously. LOC cannot be used in a relocatable program.

BEGI (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol BEGI expression

where the symbol is optional. It creates a new location counter or it redefines the value of any location counter before the counter has been used. BEGI gives the new or redefined location counter the value of the expression, but has no effect on the current location counter. BEGI cannot redefine the value of any location counter that has been used for location assignment. Any symbol used as a variable field expression must have been defined previously.

USE (DAS 4A, DAS 8A)

This directive has the format

blank USE xxxx

where xxxx is a blank, COMN, SYOR, or a user-created location counter label. It uses the location counter xxxx to assign locations to data and instructions (except literals and indirect pointers). If xxxx is PREV, the previously used location counter is recalled with the restriction that only the last-used counter can be so recalled.

Data Definition Directives

These directives control the sign and assignment of data words. In the descriptions, item refers to a data item, which can be an expression or a direct or indirect address. When there are an odd number of characters in an alpha constant, the last word is left justified and blank filled.

DATA (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol DATA expression(s)

where the symbol is optional. DATA assigns the symbol to the memory address of the first generated word. In the absence of a symbol, an unlabeled block of data is generated.

PZE (DAS 4A, DAS 8A, DAS MR)

This directive (PZE = plus zero) has the format

symbol PZE expression(s)

It is similar to DATA except that the sign bits of data words are always zeros (plus).

MZE (DAS 4A, DAS 8A, DAS MR)

This directive (MZE = minus zero) has the format

symbol MZE expression(s)

It is similar to DATA except that the sign bits of data words are always ones (minus).

FORM (DAS 8A, DAS MR)

This directive has the format

symbol FORM expression(s)

where the symbol is mandatory and the terms are absolute terms or expressions. The symbol is the name of the FORM, which specifies the format of a bit configuration of a data word. The terms specify the length in bits of each field in the generated data word, where the sum of the term values is from one to the number of bits in the computer word. FORM is ignored if there are any errors in the variable field, except that an error is flagged when a term cannot be represented in the number of bits specified when FORM is applied (by placing its name in the operation field of a symbolic source statement) to another statement. The FORM can be redefined.

Example:

BYTE	FORM	8,8
BCD	FORM	4,4,4,4
PTAB	FORM	1,2,3,4

Given the FORM definition

ABC	FORM	6,2,8
-----	------	-------

and the FORM reference

ABC	2*3,1,'A'
-----	-----------

The generated binary data word is:

0 001 100 111 000 001

Memory Reservation Directives

These directives control the reservation of memory addresses and areas.

BSS (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol BSS expression

where the symbol is optional. It increases the value of the current location counter by the amount indicated by the expression. If there is a symbol, it is assigned the value of the counter prior to such increase. The location counter always points to the next available word.

BES (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol BES expression

It is similar to BSS except that if there is a symbol, it is assigned to the address one less than the incremented location counter.

DUP (DAS 8A, DAS MR)

This directive has the formats

blank DUP n
blank DUP n,m

It duplicates source statement(s) following the DUP. The first format duplicates the next source statement n times. The second format duplicates the next m source statements n times, where $m \leq 3$ and $n \leq 32,767$.

Conditional Assembly Directives

These directives assemble portions of the program according to the conditions specified in the variable fields.

IFT (DAS 8A, DAS MR)

This directive (IFT = if true) has the format

DAS directives

blank IFT expression(s)

When a single expression is present in the variable field, the next symbolic source statement is assembled if the value of the expression is not zero. Otherwise, the next symbolic source statement is not assembled.

When two expressions are present in the variable field separated by two commas (i.e., A,,B), the next symbolic source statement is assembled if the value of the first expression is not equal to the value of the second expression. Otherwise the next symbolic source statement is not assembled.

When three expressions are present in the variable field separated by commas (i.e., A,B,C), the next symbolic source statement is assembled if the value of the first expression is less than the value of the second expression and the value of the second expression is less than or equal to the value of the third expression. Otherwise, the next symbolic source statement is not assembled.

Examples:

Operation Field	Variable Field	The next source statement is assembled
IFT	A,,B	for $A \neq B$
IFT	A,B,B	for $A \leq B$
IFT	0,A,B	for $A \leq B$ and $A > 0$
IFT	A	for $A \neq 0$

IFF (DAS 8A, DAS MR)

This directive (IFF = if false) has the format

blank IFF expression(s)

When a single expression is present in the variable field, the next symbolic source statement is assembled if the value of the expression is zero. Otherwise, the next symbolic source statement is not assembled.

When two expressions are present in the variable field separated by two commas (i.e., A,,B), the next symbolic source statement is assembled if the value of the first expression is equal to the value of the second expression. Otherwise, the next symbolic source statement is not assembled.

When three expressions are present in the variable field separated by commas (i.e., A,B,C), the next symbolic source statement is assembled if the value of the first expression is greater than or equal to the value of the second expression or the value of the second expression is greater than the value of the third expression. Otherwise, the next symbolic source statement is not assembled.

Examples:

Operation Field	Variable Field	The next source statement is assembled
IFF	A,,B	for $A = B$
IFF	A,B,B	for $A \geq B$
IFF	A	for $A = 0$
IFF	0,A,B	for $A > B$ or $A \leq 0$

GOTO (DAS 8A, DAS MR)

This directive has the formats

blank	GOTO	symbol
blank	GOTO	symbol

It skips more than one source statement and usually follows an IFF or IFT. All source statements following the GOTO up to, but not including the statement containing the symbol in its label field are deleted from the assembly process. GOTO cannot jump back to an earlier point in the program. If the first GOTO format is used, the skipped instructions are listed. If the second format (containing a comma after the symbol) is used, the listing of the skipped instructions is suppressed. This listing can also be suppressed by a SMRY directive.

CONT (DAS 8A, DAS MR)

This directive has the format

symbol CONT blank

It provides a target for a previous GOTO.

The symbol is not entered in the assembler symbol table.

NULL (DAS 8A, DAS MR)

This directive has the format

symbol NULL blank

It provides a target for a previous GOTO with the symbol entered in the symbol table. NULL has the same effect as BSS with a blank variable field.

Assembler Control Directives

These directives signal the end or continuance of an assembly.

MORE (DAS 4A, DAS 8A)

This directive has the format

blank MORE blank

It halts assembly to allow additional source statements to be put in the input device. Assembly resumes when the RUN button on the computer control panel is pressed. MORE is never listed.

END (DAS 4A, DAS 8A, DAS MR)

This directive has the format

blank END expression

It is the last source statement in the program. The expression is the execution

address of the program after it has been loaded into the computer. A blank expression yields an execution address of 00000.

Subroutine Control Directives

These directives create closed subroutines and control their use.

ENTR (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol ENTR blank

where the symbol is the name of the subroutine called. ENTR generates a linkage word of zero in the object program.

RETU* (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol RETU* expression

where the symbol is optional. It returns from a closed subroutine, generating an unconditional indirect jump to the address indicated by the value of the expression.

CALL (DAS 4A, DAS 8A, DAS MR)

This directive has the format

symbol CALL subfield(s)

where the symbol is optional and the subfields comprise a required symbol, an optional parameter list, and an optional error return list. A symbol in the label field is entered in the symbol table and assigned the present value of the current location counter. The symbol required in the variable field is the name of a subroutine. The optional lists in the variable field comprise valid data items.

DAS directives

Example:

CALL	FUNC,X,Y+1,(ERR),(GOOF)*
-------------	---------------------------------

produces a machine code identical to that obtained with

JMPM	FUNC
DATA	X,Y+1,(ERR),(GOOF)*

List and Punch Control Directives

These directives, which are operative only during the second pass of the assembler, control listing and punching during program assembly.

LIST (DAS 8A)

This directive has the format

blank	LIST	blank
-------	-------------	-------

It causes the assembler to produce a program listing. Initially, the assembler is in LIST condition.

NLIS (DAS 8A)

This directive has the format

blank	NLIS	blank
-------	-------------	-------

It suppresses further listing of the program.

SMRY (DAS 8A, DAS MR)

This directive has the format

blank	SMRY	blank
-------	-------------	-------

It suppresses the listing of source statements that have been skipped under control of conditional assembly directives. In DAS 8A, it also suppresses the listing of symbols on the first pass.

DETL (DAS 8A, DAS MR)

This directive has the format

blank	DETL	blank
-------	-------------	-------

It removes the effect of SMRY, i.e., causes listing of all source statements, including those skipped by conditional assembly directives. Initially, the assembler is in DETL condition.

PUNC (DAS 8A)

This directive has the format

blank	PUNC	blank
-------	-------------	-------

It causes the assembler to produce an object program. Initially, the assembler is in PUNC condition.

NPUN (DAS 8A)

This directive has the format

blank	NPUN	blank
-------	-------------	-------

It suppresses further production of paper tape punched with the object program.

SPAC (DAS 8A, DAS MR)

This directive has the format

blank	SPAC	blank
-------	-------------	-------

It causes the listing device to skip a line. SPAC is not listed.

EJEC (DAS 8A, DAS MR)

This directive has the format

blank	EJEC	blank
-------	-------------	-------

It causes the listing device to move to the next top of form. EJEC is not listed.

READ (DAS 8A)

This directive has the format

blank READ n,p

where n is the number of characters (20-80) from each source line to be processed by the assembler, and p is either 26 or 29 to indicate use of IBM 026 or 029 keypunch codes, respectively. Initially, the assembler processes 80 characters per line with 026 keypunch codes. If n is outside the range 20-80, the assembler resets the number of characters to 80 and outputs the error message SZ (size). If p is not 26 or 29, assembly stops with the A, B, X, and I registers equal to 026. To continue assembly, correct the card, reinsert it in the card reader, and press RUN on the computer control panel. If n is used without p, there is no comma. If p is used without n, it is preceded by a comma.

Examples:

READ 80,29	Reads 80 columns of 029 codes on succeeding cards
READ 72,26	Reads 72 columns of 026 codes on succeeding cards
READ ,29	Does not change number of columns read, but reads 029 on succeeding cards
READ 80	Reads 80 columns but does not change the code read

Unless there is an SZ error message, SMRY suppresses the listing of READ cards during the second pass.

DAS 8A Interface Directive to FORTRAN

This directive has the format

blank FORT blank

Except for comments, this is the first line of an assembly whose output is compatible with the stand-alone FORTRAN relocatable loader. In such an assembly, all two-word instructions are legal, but one-word memory-addressing instructions are limited to relative forward addressing. An expression containing symbols defined with COMM or EXT directives must be the second word of two-word instructions, or part of a DATA, PZE, or MZE directive. Literals are forbidden. Immediate instructions are recommended. The symbols and expressions are subject to the mode restrictions indicated for FORTRAN-compatible output.

Program Link Directives

These directives establish and control links among programs that have been assembled separately but are to be loaded and executed together.

NAME (DAS 8A, DAS MR)

This directive has the format

blank NAME

It establishes linkage definition points among separately assembled programs. The symbols can then be referenced by other programs. Each symbol also appears in the

Example:

	000000	R		FORT	
	000017	R		NAME	\$PE
	000000	E	\$SE	EXT	
	000000	E	\$QS	EXT	
	000000	E	\$QE	EXT	
000000	074025			STX	\$PE + 7
000001	034021			LDX	\$PE + 4
000002	054025			STA	\$PE + 9
000003	064025			STB	\$PE + 10
000004	015000			LDA	0, 1
000005	034020			LDX	\$PE + 7
000006	002000			JMPM	\$QS
000007	000000	E			
000010	000026	R		DATA	\$PE + 7
000011	014016			LDA	\$PE + 9
000012	024016			LDB	\$PE + 10
000013	002000			JMPM	\$QE
000014	000000	E			
000015	000026	R		DATA	\$SP + 7
000016	001000			JMP	0
000017	000000				
000017				ORG	*-1
000017	000000		\$PE	ENTR	
000020	002000			CALL	\$SE, 1
000021	000000	E			
000022	000001				
000023	000000			DATA	0
000024	001000			JMP	*-20
000025	000000	R			
000026	000000			DATA	0,0,0,0
000027	000000				
000030	000000				
000031	000000				
	000000			END	

label field of a symbolic source statement in the body of the program. Undefined NAME symbols cause error messages to be output.

Examples:

NAME	A
NAME	A,B
NAME	EX,WHY,ZEE

EXT (DAS 8A, DAS MR)

This directive has the format

symbol EXT symbol(s)

where both the label field and variable field symbols are optional.

It establishes linkage definition points among separately assembled programs. The symbols declare each symbol not defined within the current program. Each symbol, in both the label and variable fields, is output to the loader with the address of the last reference to the symbol. If a symbol is not defined within the current program and not declared in an EXT, it is considered undefined and causes an error message output. If a symbol is declared in an EXT but not referenced within the current program, it is output to the loader for loading, but no linkage to this program is established. If a symbol is both defined in the program and declared to be external, the EXT declaration is ignored.

Examples:

	EXT	AY
BEG	EXT	BE,SEE
	EXT	DEE,EE,EF,GEE

COMN (DAS 8A, DAS MR)

This directive has the format

symbol COMN item

where item is an absolute expression, and symbol is optional. It defines an area in blank common for use at execution time. This allows an assembler program to reference the same blank common area as a FORTRAN program. The common area is cumulative for each use of COMN, i.e., the first COMN is the base of blank common, the second COMN is the base of blank common plus the size of the first, etc.

Example:

AAA	COMN	3
	COMN	6*2
BBB	COMN	9

MOS I/O Control Directives

DAS MR accepts the MOS control directives listed below and explained in the Varian

DAS directives

620 Master Operating System Reference Manual (98 A 9952 090):

RBIN	Read binary record
RALF	Read alphanumeric record
RBCD	Read binary-coded decimal (BCD) record
WBIN	Write binary record
WALF	Write alphanumeric record
WBCD	Write BCD record
WEOF	Write end of file
REW	Rewind
SKFF	Skip files forward
SKFR	Skip files reverse
SKRF	Skip records forward
SKRR	Skip records reverse
FUNC	Function
STAT	Status
ION	I/O logical unit number

Macro Definition Directives

These directives begin and end macro definitions. The macro is the assembly equivalent of the execution subroutine. It is defined once and can then be called from the program. The macro is an algorithmic statement of a process that can vary according to the arguments supplied. It is assembled with the resultant data inserted into the program at each point of reference, whereas the subroutine executed during execution time appears but once in a program. Its definition comprises the statements between MAC and EMAC.

MAC (DAS MR)

This directive has the format

symbol MAC blank

and introduces a macro definition. The symbol is the name of the macro.

EMAC (DAS MR)

This directive has the format

blank EMAC blank

and terminates the definition of a macro.

A macro is called by the appearance of its name in the operation field of a symbolic source statement. The variable field of this statement contains expression(s), P(1),P(2),...,P(n), that are evaluated and stored in a table within the assembler. The macro definition is then processed with the values in the table being substituted for the respective values of the expressions in the source statement variable field. For example, if the variable field of the symbolic source statement contains:

2,B,9 + 8, = 63

then within the generated macro, P(1) = 2, P(2) is the value of B, P(3) = 021, and P(4) is the address of the value 63. All terms and expressions within the macro-referencing symbolic source statement parameter list are evaluated prior to calling the macro.

If the label field of such a source statement contains a symbol, the symbol is assigned the value and relocatability of the location counter at the time the macro is called but before data generation.

A macro definition can contain references to machine instruction mnemonics or to assembler directives other than DUP. Macros can be nested within macros to a depth limited only by the available memory at assembly time.

Example:

a. Definition of the macro:

SBR	MAC	
	SEN	0200 + P(1),* + 3
	JMP	*-2
	EMAC	

b. Calling the macro:

SBR 031

c. Expansion of the macro:

SEN 0231,*+3
JMP *-2

The parameter P(0) can also be referenced within the macro. P(0) is the first entry in the table formed by the assembler and contains the number of entries in that table, i.e., the number of parameters on the call line. This example shows the output listing obtained by calling P(0):

		1	A	MAC
		2	DATA	P(0)
		3	EMAC	
00001	00000	A 4	A	
00002	00001	A 5	A 1	
00003	00002	A 6	A 1,2	
00004	00003	A 7	A 1,2,3	
00005	00004	A 8	A 1,2,3,4	
00006	00005	A 9	A 1,2,3,4,5	
		10	END	

Relocatability Rules

A relocatable program is one that has been assembled with its instruction locations assigned in such a manner that it can be loaded anywhere in memory and executed without problems. When such a program is loaded, the beginning memory address is specified, and a value (known as the reloca-

tion bias) is added to the addresses of subsequent relocatable instructions. The programs are usually assembled with a zero relocation bias on the first instruction.

The location counter contains the (relative) address of the instruction currently being executed. The location counter is absolute when it contains the actual address of the instruction during execution, and relocatable when it contains the relative address of the instruction during execution (the current address minus the address of the start of the program).

Symbols can be absolute or relocatable. Expressions, since they contain symbols, can be absolute or relocatable. Constants are always absolute.

Figure 21-6 shows, for each arithmetic operation, whether the result is absolute (abso), relocatable (relo), or illegal.

The loader can load a relocatable program in any area of memory and modify the addresses as it loads so that the resulting program executes correctly. Programs can contain absolute and relocatable addresses and absolute and relocatable data. At the beginning of each assembly, the location counter is set to zero, relocatable. It is incremented after every instruction word or data word generated by the assembler. It can be set by the ORG directive. If an ORG directive is encountered the location-

	A is abso, B is abso,	A is abso, B is relo,	A is relo, B is abso,	A is relo, B is relo,
A+B	abso	relo	relo	illegal
A-B	abso	illegal	relo	abso
A*B	abso	illegal	illegal	illegal
A/B	abso	illegal	illegal	illegal

Figure 21-6. Results of Arithmetic Operations

counter is made absolute if the corresponding expression is absolute, or relocatable if the corresponding expression is relocatable.

If a symbol is equated to the location counter, it is relocatable if the location counter is relocatable. Otherwise, the symbol is absolute.

Source Statement Formats

Punched Card Format

Punched cards used as input to the DAS assembler contain four fields: label, operation, variable, and comment.

The label field is in columns one through six. Its use is optional. It assigns a symbol, extended symbol, or target number to a symbolic source statement.

The operation field is in columns eight through 14. It contains a mnemonic for a machine instruction or an assembler directive. Indirect addressing is specified by an asterisk following the mnemonic.

The variable field begins in column 16 and ends with the first blank not contained in a character constant. Its use depends on the instruction or directive. The variable field contains zero or more subfields, but if two or more subfields are present, they are separated by commas.

The comment field fills the remainder of the card. If there is a blank variable field, the comment field begins in column 17. The assembler ignores this field, but its contents appear in the output listing.

An asterisk in column one indicates that the entire card contains a comment. The assembler ignores the card, but its contents appear in the output listing.

Paper Tape Format

Paper tape used as input to the DAS assembler contains source statements of up to 80 characters each (not including the carriage return and line feed characters). Each statement contains four fields: label, operation, variable, and comment. The first three are separated by commas, and the comment field starts after the first variable field blank that is not part of a character constant. At the end of each statement is a carriage return (CR), followed by a line feed (LF).

The label field contains a symbol, extended symbol, or target number. If the first nonblank statement in the statement is a comma, the label field is blank.

The operation field contains a mnemonic. An asterisk following the mnemonic specifies indirect addressing.

The variable field can be blank, or contain one or more subfields separated by commas. Each subfield can contain a constant or expression, depending on the instruction or directive, or it can be voided by using adjacent commas. The variable field terminates with a blank that is not part of a character constant, or with a CR or LF.

The comment field fills the remainder of the statement from the terminating blank of the variable field to the next CR or LF. The assembler ignores this field, but its contents appear in the output listing.

If the first nonblank character of the statement is an asterisk, the entire statement is a comment. The assembler ignores the statement, but its contents appear in the output.

Assembler Output Listing

DAS Source/Object Listing

The listing can be obtained in whole or in

part as the program is being assembled. The source (symbolic) program and the object (absolute) program are listed side-by-side on the listing device output paper. This device is either a Teletype or a line printer.

The listing is output according to the specifications given by the list and punch control directives in the assembly.

Error analysis during assembly causes the

error messages described in the following section to be output on the line following the point of detection.

The following example illustrates the format of the outlisting. In addition, on DAS MR listings, a line count appears. The addressing modes are: FORTRAN common reference = C, externally defined = E, indirect pointer = I, and absolute or relative = R.

Example:

Location	Code	Mode	Source Statement
014000			ORG 014000
014000	000000		ABS ENTR
014001	001002		JAP* ABS
014002	114000	R	
014003	005211		CPA
014004	001000		JMP* ABS
014005	114000	R	
	000000		END

DAS Error Messages

The assembler checks syntax on both passes. During pass 1, detectable errors are listed. During pass 2, the following information is listed:

- Error code
- Value of location counter
- Object code when the instruction is assembled.

This code is suppressed by NLIS directives and list-suppression commas in GOTO directives.

The error message appears in the listing line following the statement found in error. Each line can hold up to four error messages. Figure 21-7 lists the error codes recognized by DAS, and their meanings.

DAS 4A I/O Operations

Load the program into memory using the binary loader. Execute it by entering a positive, nonzero value in the A register during loading, or by clearing all registers and pressing RESET and RUN after the loading.

During execution, the program first determines the amount of memory required. It then stores in memory location 00003 a value one less than the lower limit of the binary load/dump. This is the highest address that the assembler section can use without destroying part of the binary load/dump.

I/O Device Definition

The program then makes three requests for definitions of I/O devices:

ENTER DEVICE NAME FOR XX

Code	Meaning
*AD	An address expression is in error.
*DC	A decimal character appears in an octal constant;
*DD	There is an invalid redefinition of a symbol or the location counter.
*E	The symbolic source statement is incorrectly formed.
*EX	An expression contains an illegal construction.
*FA	A floating point number contains a format error.
*IL	The first non-blank character of the source statement is invalid; the statement is not processed.
*NR	No memory space available for the addition of an entry to the assembler's tables.
*NS	No symbol in the label field of a SET, EQU, MAC or FORM directive statement. No symbol in the label or variable field of an OPSY directive statement. No symbol in the variable field of a NAME directive statement.
*OP	The instruction field is undefined; two No Operation (NOP) instructions are generated in the object program. The remainder of the statement is not processed. Illegal nesting of DUP or MAC directive statements also cause this error.
*QQ	Illegal use of prime (').
*R	A relocatable item was encountered in the place where an absolute item was expected.
*SE	The symbol in the location field has a value during pass 2 that is different than the value used in pass 1.
*SY	An expression contains an undefined symbol.
*SZ	The expression value is too large for the size of the subfield or a DUP statement specifies that more than three symbolic source statements are to be assembled.
*TF	Undefined or illegal index register specification.
*UC	An undefined character in an arithmetic expression.
*UD	Undefined symbol in variable field of USE directive.
*XR	Address out of range for indexing specification.
*=	Illegal use of a literal.

Figure 21-7. DAS Error Codes

where xx is one of the I/O function names SI (source input), BO (binary output), or LO (list output), respectively. Respond to each request in turn by typing the name of the desired device and a carriage return (CR). Figure 21-8 lists the acceptable device names in response to each request. If the default assignment is desired, merely press CR. If the device name is incorrect, the message

DEVICE NAME NOT VALID

is output and the request repeated.

Output of any line to the TTY can be terminated by pressing RUBOUT. The error correction feature (page 21-26) can be used during specification of I/O devices to the program.

When I/O assignments are complete, the I/O section uses the binary loader to load the assembler section into memory.

To restart the I/O section before the assembler section is loaded, press STEP,

clear all registers, and press RESET and RUN.

The following sections describe the subroutines applicable to each type of I/O function.

Source Input I/O Subroutines

The following I/O devices can be source program input devices. The I/O subroutines to communicate with these devices are in the I/O section.

- a. 33/35 ASR TTY
- b. 33/35 ASR TTY paper tape reader
- c. High-speed paper tape reader
- d. Card reader

The **TY subroutine** communicates with the 33/35 ASR TTY. The (effective) calling sequence is:

Assembly Function	Device	Default Assignment
SI (source input)	TR (TTY paper tape reader)	TR
	TY (TTY keyboard)	
	PR (high-speed paper tape reader)	
	CR (Model 620-22, 23 and 25 card readers)	
	CR1 (Model 620A card reader, micro verse logic)	
LO (list output)	TY (TTY printer)	TY
	LP (Model 620-76 line printer)	
	LP1 (Model 620-75 line printer)	
	LP2 (Model 620-77 line printer)	
BO (binary output)	TP (TTY paper tape punch)	TP
	PP (high-speed paper tape punch)	
	CP (Model 620-27 card punch)	
	CP1 (Model 620-26 card punch)	

Figure 21-8. Acceptable I/O Devices

DAS operations

JMPM	0500
DATA	WORD COUNT
DATA	ADDRESS
Return	

TY inputs characters from the TTY keyboard until either a carriage return is input or the word count (two characters per word) specified in the calling sequence is reached.

As each character is input, it is verified as an ASCII character. If it is not, it is ignored. ASCII characters are output to the TTY page printer and are stored sequentially in memory beginning at the address specified in the calling sequence.

If the specified word count is 36 or larger, 36 words are input.

After the specific number of words or a carriage return is input, a line feed and a carriage return are output to the TTY page printer, signifying that input is complete.

To delete a line, while it is being input, type a backslash (press the shift key and the letter L simultaneously). A line feed and a carriage return are output to indicate that the line has been deleted. Characters entered before the backslash become blanks.

To delete a character just entered, type a backarrow. This is printed by the TTY page printer to indicate the deletion. As many backarrows as necessary can be entered. Each deletes one character (but not beyond the beginning of the line). Characters deleted change to blanks (in the data) but are printed with their accompanying backarrows on the page printer (each succeeding backarrow indicating deletion of the preceding character), e.g.:

IDEAX ← indicates the X is deleted

BOTHER ← ← indicates that R and E
are deleted

The TR subroutine communicates with the 33/35 ASR TTY. The (effective) calling sequence is:

JMPM	0500
DATA	WORD COUNT
DATA	ADDRESS
Return	

TR inputs characters from the TTY paper tape reader until either a carriage return is input or the word count (two characters per word) specified in the calling sequence is reached.

As each character is input, it is verified as an ASCII character. If it is not, it is ignored. ASCII characters are stored sequentially in memory beginning at the address specified in the calling sequence.

After the specified number of words has been input, characters are input but not stored in memory until there is a carriage return. When a carriage return is input, TR inputs two more paper-tape characters and stores them in memory locations 4 and 5. These two characters appear in the beginning of the next record when the next call is made to input a record from the TTY paper tape reader.

The PR subroutine communicates with the high-speed paper tape reader. The (effective) calling sequence is:

JMPM	0500
DATA	WORD COUNT
DATA	ADDRESS
Return	

PR inputs characters from the high-speed paper tape reader until either a carriage return is input or the word count (two characters per word) specified in the calling sequence is reached.

As each character is input, it is verified as an

ASCII character. If it is not, it is ignored. ASCII characters are stored sequentially in memory (two characters per computer word) beginning at the address specified in the calling sequence.

After the number of words specified in the calling sequence has been input, characters are input but not stored in memory until there is a carriage return. When a carriage return is input, PR terminates input from the high-speed paper tape reader.

The CR subroutine communicates with the card reader. The (effective) calling sequence is:

JMPM	0500
DATA	WORD COUNT
DATA	ADDRESS
Return	

CR inputs columns from the card reader and converts them to ASCII characters until the word count (two characters per word) specified in the calling sequence is reached. The ASCII characters are stored sequentially in memory beginning at the address specified in the calling sequence.

Binary Output I/O Subroutines

The following I/O devices can be source program output devices. The I/O subroutines to communicate with these devices are in the I/O section.

- a. 33/35 ASR TTY paper tape punch
- b. High-speed paper tape punch

The TP subroutine communicates with the 33/35 ASR TTY. The (effective) calling sequence is:

JMPM	01300
------	-------

DATA	WORD COUNT
DATA	ADDRESS
Return	

TP outputs data to the TTY paper tape punch in a format compatible with the binary loader. The data are output from memory beginning at the specified address. The number of data words to be output is also specified in the calling sequence.

The binary loader requires three paper-tape frames for each data word. The most significant four bits are contained in the first frame; the next six bits in the second, and the least significant six bits in the third.

Each frame of paper tape can contain eight bits of information. Frame bit positions 1 through 6 contain the data. Frame bit position 7 contains a check bit that is the inverse of position 6. Frame bit position 8 is zero.

If the specified data word count is negative, 100 frames of paper tape are punched with frame bit position 8 punched and positions 1 through 6 not punched (leader/trailer). Any entry in the data address portion of the calling sequence is ignored.

The PR subroutine communicates with the high-speed paper tape punch. PP operates in the same manner as TP.

List Output I/O Subroutines

The following devices can be list output devices. The subroutines to communicate with these devices are included in the I/O section.

- a. 33/35 ASR TTY page printer
- b. Line printer

The TY subroutine communicates with the

33/35 ASR TTY. The (effective) calling sequence is:

JMPM	01000
DATA	WORD COUNT
DATA	ADDRESS
Return	

TY outputs ASCII characters to the TTY page printer. They are packed two per computer word and are output from memory beginning at the specified address.

If the specified word count is 36 or larger, 36 words are output.

When the specified number of words has been output, a line feed and a carriage return are output.

If the specified data word count is negative, nine line feeds and a carriage return are output to position the paper to the top of the next page. Any entry in the data address portion of the calling sequence is ignored.

The LP subroutines communicate with the line printer. The (effective) calling sequence is:

JMPM	01000
DATA	WORD COUNT
DATA	ADDRESS
Return	

LP outputs ASCII characters to the line printer. The characters are packed two per computer word and are output from memory beginning at the specified address. If the specified word count is 66 or larger, 66 words are output.

Single-line slewing (paper movement) is supplied with each line output.

If the specified data word count is negative, the next line is printed at the top of the

next page. Any entry in the data address portion of the calling sequence is ignored.

Assembler Section

When control is given to the assembler by the binary loader, it halts with a one in the program counter (P register). The operator should set SENSE switch 1 ON if an assembly pass 1 is desired, or SENSE switch 1 OFF if an assembly pass 2 is desired.

If an assembly pass 1 is selected, the operator should ready the source input I/O device (SI) with the symbolic source input statements and press RUN. The assembler proceeds to input the symbolic source statements and performs a pass 1 assembly. When the symbolic source END statement (denoting the completion of the pass 1 assembly process) is assembled, the assembler halts with a one in the program counter (P register). The operator should again select the desired assembly pass as described above.

If an assembly pass 2 is selected, the operator should ready (SI) with the symbolic source input statements, ready the binary output (BO) and list output (LO) I/O devices, and press RUN. The assembler proceeds to input the symbolic source statements and performs a pass 2 assembly outputting binary object loader text and an assembly listing. The binary output can be suppressed by setting SENSE switch 3 ON. The assembly listing output can be suppressed by setting SENSE switch 2 ON. Error diagnostics cannot be suppressed and are output on LO as they are detected during pass 2. It should be noted that SENSE switches 2 and 3 are only interrogated during pass 2 and have no effect during pass 1.

Pass 2 can be restarted or repeated without repeating pass 1.

At the completion of the pass 2 assembly process, the assembler halts with a one in the program counter (P register). The assembler is now ready to accept another assembly with the same I/O devices as previously specified. If it is desired to change the I/O devices, it is necessary to reload, starting with the I/O section as outlined previously.

If it is necessary to restart the assembler, press STEP, set all registers to zero, and press computer RESET and RUN. The assembler halts with a one in the program counter (P register) and is now ready to accept another assembly.

A MORE directive causes the computer to stop and wait until the input units are prepared and RUN is pressed. MORE is indicated by 0170017 in the A register.

Synchronization errors detected on pass 2 indicate that the value of a label on that pass does not agree with the value it had on pass 1. Such errors are due to source tape misreadings. They halt the assembly, as indicated by 0777 in the A register. To continue, press RUN. The assembler resets the location counter value to that assigned during pass 1, prints the error message *SE, and continues.

DAS MR Operations

DAS 8A Operations

Load the program into memory using the binary loader. Prepare the system input (SI), system output (SO), and list output (LO) units, and set the SENSE switches as follows. On pass 1, set SENSE switch 1 only. On pass 2, reset SENSE switch 1 and, for listing output, set SENSE switch 2, or, for binary object output, set SENSE switch 3. To begin assembly, RUN at address 000001.

END terminates both passes 1 and 2 by executing a HALT. After pass 1 is terminated, begin pass 2 by resetting the I/O devices, setting the SENSE switches as above, and pressing RUN. To obtain extra copies of the program, repeat pass 2 as desired. END is indicated by 017777 in the A register.

Since DAS MR is used within MOS and uses the MOS I/O control system, the I/O devices can be defined as required (see the Varian 620 Master Operating System Reference Manual).

DAS MR inputs the symbolic source statements from the processor input logical unit (PI) in alphanumeric mode, performs a pass

1 assembly, and outputs them in the same mode on the processor output logical unit (PO). When END is detected, pass 1 terminates, the assembler backs up the number of output symbolic source statements, and begins pass 2. This pass inputs the state-

DAS operations

ments from the system scratch logical unit (SS), produces a listing on the LO unit, and a binary object program on the BO unit. Note that PO and SS must be the same magnetic tape, drum, or disc unit.

If a listing is not wanted, use the following directive to the MOS executive when requesting the assembly:

```
/ASSEMBLE      N
```

If a binary object program is not wanted, use the following directive:

```
/ASSEMBLE      B
```

If the memory map portion (symbol table, external names, and entry names) of the assembly listing is not wanted, use the following directive:

```
/ASSEMBLE      M
```

To read the same physical symbolic source statements for both assembly passes, use the following directives to the MOS executive:

```
/ASSIGN      PO = DUM,SS = PI
/ASSEMBLE
```

The PO symbolic source statement output serves as a copy of the statements, and can be the input for another assembly.

Memory Size Assembler Version	4K	8K	>8K
DAS 4A	~ 150	~ 1450	~ 1450 + n (1300)
DAS 8A	—	~ 440	~ 440 + n (800)
DAS MR	—	~ 20	~ 20 + n (800)

n = number of 4K memory increments above 8K

Figure 21-9. Symbol Table Capacities

SECTION 22 – BINARY LOAD/DUMP PROGRAM (BLD II)

The binary load/dump program allows the user to load object programs from a paper tape or TTY reader, or to punch the binary contents of memory on paper tape in a reloadable format.

Location of BLD II

BLD II is loaded using the manual bootstrap routine or the automatic bootstrap loader (ABL) option. Once loaded into memory, BLD II relocates itself into the upper part of the highest 4K memory module unless the operator specifies a different 4K memory module.

Initially, BLD II occupies addresses 07000 to 07755. By residing in these locations, it does not interfere with the bootstrap loader occupying addresses 07756 to 07776. Immediately after loading, BLD II relocates to occupy addresses 0x7400 through 0x7755. x denotes the 4K memory Module in which BLD II relocated as follows:

X =	Memory Module
0	4K
1	8K
2	12K
3	16K
4	20K
5	24K
6	28K
7	32K

Entry to BLD II to read object tapes is always 0x7600

0 x x x 1 1 1 1 1 0 0 0 0 0 0

and entry to punch object tapes is 0x7404.

0 x x x 1 1 1 1 0 0 0 0 1 0 0

Program Options Prior to Loading

The operator has four options prior to loading the binary load/dump program tape.

- a. No SENSE switches set: The program accepts input from the device specified in the entered bootstrap routine and stores the program in the highest 4K memory module. After reading the program in, the computer halts with the P register set to the entry address (0x7600) of the binary loader routine and the A, B, and X registers cleared.
- b. SENSE switch 1 set: This allows the operator to select any 4K memory module in which the program is to operate. After reading the program in, the computer halts with the P register set to 07013. The operator must enter in the A register the number (0 through 7) specifying the memory module in which the program is to reside. By pressing RUN, the operator initiates the relocation and the computer halts as in item (a) above.
- c. SENSE switch 2 set: This adjusts the dump routine for TTY punch output.

- d. SENSE switch 3 set: This allows the operator to splice an object program to the BLD II program tape, load the BLD II program and the object program and execute the object program without further intervention.

Object Program Loading Options

The binary load/dump program establishes the input reader (TTY or high-speed paper tape) and the output punch devices by interrogating the bootstrap loader. For this reason, the binary load/dump program is adjusted to accept object program tapes from only the reader specified by the bootstrap routine.

However, setting SENSE switch 2 prior to loading the load/dump program adjusts the program for TTY punch regardless of the bootstrap-routine-specified I/O devices.

Punching Tapes of Memory Contents

To punch a tape from memory to the high-speed paper tape punch, SENSE switch 2 must not be set when the load/dump program is entered. To punch a tape from memory to the TTY punch, SENSE switch 2 must be set when the load/dump program is entered if the input reader is a high-speed paper tape device. The operator can specify that tapes be punched in binary format to load using the binary loader or punch the binary loader in bootstrap-loadable format.

To punch a tape in binary format:

- a. Set the P register to 0x7404.
- b. Set the A register to the address of the first word to be punched.
- c. Set the B register to the address of the last word to be punched.

- d. Set the X register to the address of the first instruction to be executed at load time. (To punch non-contiguous areas, set the X register to -1(0177777) for all but the last area to be punched.)

To punch the BLD II program in a bootstrap-loadable format, set the P register to 0x7400. Set the A and B registers to zero and the X register to non-zero.

Loading the Binary Load/Dump Program

- a. Enter the bootstrap loader
- b. Clear the I register.
- c. Set the P register to 007770.
- d. Set the X register to 007000.
- e. Set SENSE switches, if required.
- f. Turn on the paper tape reader. (If ASR 33/35 is used, turn TTY to local and press CTRL D, T and Q, then turn back on line.)
- g. Position the BLD II tape in reader with the first data frame after the eight-level punches under the high-speed reader head or under the read station of the TTY reader. (Place reader control lever of TTY to stop position.)
- h. Set STEP/RUN to RUN and press START. (If ASR 33/35, press START; then set the control lever to RUN position.) Load is complete when the computer changes to step mode.
- i. If SENSE switch 1 is set, reset SENSE switch 1 and clear the A register.
- j. Enter the appropriate octal value in the A register to relocate load/dump program, if applicable. Press START.

- k. The computer will halt in step mode with the P register containing 0x7600 unless SENSE switch 3 was set. With SENSE switch 3 set, the computer will execute the object program.
- l. Remove the BLD II program tape and reset SENSE switch 2, if applicable.

Loading the Object Program Tape

Object program tapes can be loaded immediately after the binary load/dump program because the P register is set to the load starting address (0x7600). For all subsequent loadings, assure that P is set to 0x7600.

Verification

To ensure that an object tape contains no errors before loading into core memory, the binary load/dump program has an option that performs only check-sum error-checking. To use this option:

- a. Turn on the reader and position the object tape in the reader.
- b. Set P register to X7600.
- c. Enter 100000 in the A register.
- d. Clear the I register.
- e. Set STEP/RUN to RUN and press START
- f. A verify with no errors is indicated by the computer halting with:

P register = 0x7600
 A register = 100000
 B register = 000000
 X register = execution address

- g. A verify with a check-sum error is indicated by the computer halting with:

P register = 0x7600

A register = 100000

B register = 177777

X register = load address of last correct address read.

- h. To retry the check-sum error record, reposition the object program tape at the previous visual aid and press RUN. If a check-sum error is read again, check each character in the record. An error has been made in punching or the tape may be slightly torn.

Load Program and Halt

To load the object program and halt:

- a. Turn on the reader and position the object tape in the reader.
- b. Clear the A, B, X, and I registers.
- c. Set the P register to 0x7600.
- d. Set STEP/RUN to RUN and press START.
- e. Correct loading will be indicated by:

P register = 0x7600

A register = 000000

B register = 000000

X register = program execution address

- f. A check-sum error is indicated by the same conditions as above.

Load the Program and Execute

Programs can be loaded and executed using the steps in the previous section except, in step b, set the A register to 000001 (any positive number).

Punching Program Tapes

With the binary load/dump program loaded and paper tape punch on, areas of memory can be punched in object-tape-loadable format.

- a. Set the A register to the beginning address of area to be punched.
- b. Set the B register to the last address to be punched.
- c. Set the X register to the address of first instruction to be executed at load time, or if noncontiguous memory areas are to be punched, set the X register to -1 (177777).
- d. Set the P register to 0x7404.
- e. Clear the I register.
- f. Set STEP/RUN to RUN and press START.
- g. Tape will be punched and the computer halts with registers unaltered. If additional areas are to be punched, perform steps a through f above, entering the new areas in the A and B registers. Prior to punching the last area, set the X register to the address of the first instruction to be executed at load time.

SECTION 23 – DEBUGGING PROGRAM (AID II)

Varian provides the AID II on-line debugging program for all Varian 620/f systems. AID II is the most advanced of the debugging aid packages for the Varian 620 family of computers.

AID II provides the user with software to facilitate program checkout. By entering AID commands on the TTY, the operator can display and alter the contents of any memory address or block. Programs can be trapped into and out of user-selected blocks and can be searched for specific conditions. Any program can be entered, monitored, and altered from the TTY.

As an added feature, data can also be dumped onto magnetic tape, punched out on paper tape, or printed on the TTY keyboard. Object programs can thus be converted from one media to another, simply and directly.

Loading AID II

AID II is loaded using the binary load/dump routine. Once loaded, AID II is positioned in the memory addresses just below the binary load/dump routine. It always occupies the highest memory addresses regardless of memory size. AID II occupies X6000-X7377 memory addresses.

Register and Memory Modification

With AID entered and the computer in the run mode, the following entries on the TTY keyboard produce the indicated results. Entering a RUBOUT character cancels entries, or if an operation is in progress, terminates it.

Operator Enters:	Operation and Results
A.	The program types the contents of the pseudo-A register on the TTY keyboard. If the contents are to be changed, type the desired number and a period; otherwise, type only a period.
B.	Same as above for the pseudo-B register.
X.	Same as above for the pseudo-X register.
Cxxx.	Same as above for memory contents xxx. By typing a comma (,) instead of a period, the operator requests the next memory address for display.
Gxxx.	Loads the contents of the pseudoregisters into the respective A, B, and X registers and starts execution at address xxx.
Vxxx.	The program starts typing memory address contents at xxx and continues until a RUBOUT character is entered. The left column is the base address in octal. The contents of eight memory addresses are typed across the page in ascending order. The first number in the next line indicates the base

address for the next eight memory address contents.

Sx,y,z,m.

Search through memory starting at x and ending at y for the value of z masked by value of m. A masked-search searches for comparison with value of z for each bit corresponding to a one in the m value. Each time the values compare, the address and value are printed on the TTY. By typing an N instead of a mask value, the program searches for the negative value of z. Omission of m assumes an all-ones mask.

Ix,y,z.

The program stores the value of z in all memory addresses starting at address x and ending at address y.

Preset registers and go to location x. If and when location x is reached, save and type location y, the contents of y, and the current values of the registers. (Trap to y from x.)

Ty,

Continue trap from last breakpoint location to new breakpoint location y. (Present and save registers as before.)

WARNING: An incomplete trap (no return to AID II) will leave the object program with the instruction at the breakpoint location changed to a return jump to the AID II trap return. These locations should be restored to their original values before further use of the trap function to ensure proper results.

Handling Paper Tape

The paper tape reader and punch can be controlled through AID II for some operations. With AID II entered and the computer in run mode, the paper tape system responds as indicated.

Operator

Enters:

Operation and Results

Dx,y,z,.

Punch a program tape from the contents of address x to the address y, with execution address z.

X = -1 allows punching of non-contiguous areas.

Lm,

If the value m is one and no check-sum errors were encountered, the program is executed. If the value of m is zero and no check-sum errors were encountered, the TTY prints the contents of the A, B, and X registers, respectively, on the next line. The printout for the A and B registers will be zeros. The X register printout is the execution address of the object program. If a check-sum error is encountered, the B printout contains -1, and the X printout contains the address of the last record from the tape. If M = -1 the operation is the same as zero except the object tape is verified but not loaded into core.

Magnetic Tape Handling

Data can be manipulated from and to magnetic tape through AID II. With AID II

entered and the computer in run mode, the magnetic tape responds as indicated.

Operator

Enters: Operation and Results

Eu. Write a file mark on magnetic tape unit u.

Fn,u, Skip to file n on magnetic tape unit u.

N, Skip to the next file on the magnetic tape unit previously designated.

Pu, Backspace one record on magnetic tape unit u.

Ru, (or Ru.) Read an object magnetic tape into memory from magnetic tape unit u. A

period causes a load and return to AID II. A comma causes a load and execute. If an uparrow is printed, a file mark was read on the tape. If an octal number is typed out, a parity error was encountered and the address of the error is indicated.

Wx,y,z,u

Write an object magnetic tape from memory where x is the starting address, y is the ending address, z is the execution address, and u designates the magnetic tape unit.

SECTION 24 – SOURCE PROGRAM EDITOR (EDIT)

EDIT is a standard program that permits the programmer to prepare and write symbolic programs and generate a symbolic program tape. EDIT is very flexible in that the symbolic program can be typed, on line, on the teletype keyboard and stored directly in the core memory. Then, using EDIT commands, the program can be listed (printed).

EDIT also adds, corrects, and deletes any portion of the symbolic program. When the program is correct and ready to be assembled or compiled, EDIT can generate a symbolic program tape of the stored program.

EDIT is on punched paper tape in binary format, and can be loaded into memory using BLD II (in the load-and-go mode).

Using the notation

H = high speed paper tape system

T = teletype paper tape system

answer the following questions printed on the teletype.

SOURCE PAPER TAPE PROGRAM

INPUT DEVICE (H OR T)

(Type an H or a T.)

OUTPUT DEVICE (H OR T)

(Type an H or a T.)

EDIT will dynamically adapt to this configuration and enter the instruction mode. (A carriage return, line feed and asterisk will be typed. EDIT can be restarted at any time by setting the instruction and P registers to zero and pressing RUN. But it can only be re-configured on loading.

Initially, EDIT is in the instruction mode, i.e., it can accept instructions. Anything typed by the user is interpreted as a command to EDIT, but EDIT accepts only legal instructions. EDIT ignores other inputs and types a question mark.

When not in the instruction mode. EDIT is in test mode, i.e., all characters input from the keyboard or tapes are interpreted as text to be put into the text buffer in the manner specified by a preceding EDIT instruction. Figure 24-1 illustrates the transfer of EDIT from one mode to the other.

Search and Find

A very convenient feature available with EDIT is the SEARCH feature, which searches a line of text for a specified character. When you type a line number followed by S, EDIT waits for the input of the character for which it is to search. When EDIT locates the character and types it, typing stops. You can then input all or any combination of the following:

- A. New text (terminate the line with the return key).
- B. Rubout to delete the entire line to the left.
- C. Return to delete the entire line to the right.
- D. ← to delete from right to left one character for each ←
← typed (A — is echoed for each ← typed.

EDIT

E. CTRL/CC to search for the next occurrence of the search character.

F. CTRL/bell to change the search character to the next character typed by the programmer.

Another feature is the FIND that searches for a string of characters in the text. When you type F (followed by return) and a string of characters, terminated by a return, the text is searched for that string. When it is found, the line in which it first appears is typed and the current line pointer is set to that line. The program returns to instruction mode. If the string did not appear, no action is taken and the program returns to instruction mode.

Error Detection

EDIT checks all instructions for nonexistent information and incorrect formatting. When an error is detected, EDIT types a question mark and ignores the instruction.

Special Keys and Instructions

EDIT is controlled by the use of special keys and instruction. Certain keys can be used in either instruction or text mode with the mode of operation determining the function of each key as shown in Figure 24-2.

EDIT instructions are given in instruction mode. There are three basic types of instructions — input, editing, and output (Figure 24-3). Instructions are executed when return is pressed.

Instruction	Meaning
A	Add incoming text from the keyboard to the text buffer immediately following the text currently etched in the buffer.
R	Read incoming text from the tape reader and append it to the text currently stored in the buffer.
L	List entire text buffer. Specify one line or a group of lines, including line numbers.
C	Change a line. Precede the instruction with the decimal line number or line numbers of the lines to be changed.
I	Insert into text buffer. Specify the line where the inserted text is to begin.
D	Delete from text buffer. Specify the line or group of lines to be deleted.
P	Punch text buffer. Specify one line, a group of lines, or the entire text buffer.

All instructions are executed when the return key is depressed. Any other character causes the instruction to be ignored and a question mark typed.

Figure 24-1. Transfer between EDIT Modes

Instruction Mode	Text Mode
Execute preceding instruction	Enter line in text buffer
Illegal	Cancel line to the left margin
Cancel preceding instruction	Delete to the left one character for each depression (A is echoed each time)
Respond with * and remain in instruction mode	Return to instruction mode and print
Value equal to decimal	Legal text character
Value of current line (used alone or with — and A number, E.G., — B	
Value equal to number of last line in buffer (used as an argument)	Legal text character
List next line	
Used with or to detain their value	
	Produces a tab interpreted as 7 spaces or output

Figure 24-2. EDIT Key Functions

Instruction	Function
Input	
A	Add incoming text from keyboard to text buffer.
R	Add incoming text from paper tape reader to text buffer
Editing	
L	List entire text buffer
NL	List line N
M,NL	List lines M through N inclusive with line numbers
NC	Change lines M through N inclusive

Figure 24-3. EDIT Instructions (1 of 2)

M,NC	Change lines M through N inclusive
I	Insert before first line
NI	Insert before Line N
K	Delete entire text buffer
ND	Delete line N
M,ND	Delete lines M through N inclusive
G	Print next line beginning with an alphabetic character (if none, no action taken)
NG	Print next line after N beginning with an alphabetic character (if none, no action taken)
S	Search buffer for character specified after return. Allow modification (search character is not echoed on printer), and print lines while searching
NS	Search line N, as above
M,NS	Search M through N inclusive, as above
F XXXX	Search text for string XXXX (up to 72 characters), and type out line in which it appears (if string not found, return to instruction mode)
NF XXXX	Search from line N as above
Output	
P	Punch entire text buffer
NP	Punch line n
M,NP	Punch lines M through N inclusive
T	Punch about 20 inches of leader/trailer tape

Note: M and N are decimal integers, with $M \leq N$.

Figure 24-3. EDIT Instructions (2 of 2)

SECTION 25 – MATHEMATICAL PACKAGE

This section is intended to acquaint the programmer with the standard subroutine library. The library is detailed in Varian Subroutine Descriptions (98 A 9402 041).

If a subroutine requires only one parameter or argument, programmed entry is made by first loading the desired parameter into the A register, or A and B registers for a double or floating point entry, and then executing a jump and mark instruction to the subroutine.

Where more than two input parameters are required, the parameters are entered into the program following the jump to the subroutine. The following sequence of instructions are used.

Fixed-point single-precision multiply

Identification: XMUL.

Provides the software version of the (optional) hardware multiply instruction. Uses recursive addition of multiplicand with shifting.

Fixed-point single-precision divide.

Identification: XDIV.

Provides the software version of one (optional) hardware divide instruction. The true remainder and quotient are delivered to the A register and B register, respectively.

Location	Instruction	Remarks
P	Jump and mark	Jump to subroutine.
P + 2	Parameter	Parameters or parameter locations for subroutine.
P + 3	Parameter	Parameters or parameter locations for subroutine.
P + 4	Parameter	Parameters or parameter locations for subroutine.
P + n	Parameter	Parameters or parameter locations for subroutine.
P + n + 1	Normal return	Continuation of program.

Figure 25-1. Sequence of Instructions for Entering Multiple Parameters

Fixed-point double-precision 2's complement

Identification: XDCO.

Takes the 2's complement of the double-precision number in the A register and B register. The X register is unchanged. The argument is complemented and the low-order bits are tested for a carry condition.

Fixed-point-double-precision add

Identification: XDAD.

Adds a double-precision number whose high-order address is in the calling sequence to the double-precision numbers in the A register and B register. The X register is unchanged. Low-order words are added first and any carry generated is added to the high-order sum.

Fixed-point double-precision subtract

Identification: XDSU.

Subtracts a double-precision number whose high-order address is in the calling sequence from the double-precision number in the A register and the B register. The X register is unchanged.

Fixed-point double-precision multiply

Identification: XDMU.

Multiplies the double-precision number whose high-order address is in the calling sequence times the double-precision number in the A register and the B register. The X register is unchanged. Uses double-precision addition of partial products.

$$(A + a \cdot 2^{-15}) (B + b \cdot 2^{-15}) = AB \cdot 2^0 + Ab \cdot 2^{-15} + aB \cdot 2^{-15}$$

Fixed-point double-precision divide

Identification: XDDI.

Divides the double-precision number in the A register and B register by the double-precision number whose high-order address is in the calling sequence. The X register is unchanged.

Absolute value, floating point (type real)

Identification: ABS.

Takes the absolute value of the floating-point (real) quantity in the A, B registers, returning the result to the A, B registers. The absolute value of a is defined as $-a$ if a is negative, as a if a is not negative.

Absolute value, fixed point (type integer)

Identification: IABS.

Takes the absolute value of the 16-bit signed integer in the A register and returns the result to the A register. The absolute value of a is defined as $-a$ if the a is negative and if a is non-negative.

Transfer of sign, fixed point (type integer)

Identification: ISIGN.

Applies the sign of the called (second) parameter to the quantity in the accumulator (first parameter). The parameters and result are fixed point quantities. Uses \$SE.

Copy sign

Identification: SIGN.

Sets sign of floating point number equal to that of argument. Output in A, B registers. Uses \$SE.

Separate mantissa

Identification: \$FMS, \$FSM.

Separates a positive floating point number into characteristic and mantissa. Output in A, B (mantissa) and X (characteristic) registers.

Floating point number to integer

Identification: \$HS.

Converts a floating point number to an integer. Uses \$SE.

Normalize.

Identification: \$NML.

Normalizes a double precision number. Shifts to sign and tests for sign set. Uses XDCO. Output in A, B registers. Flag for sign in X register.

Floating add

Identification: \$QK.

Algebraically adds 2 floating point numbers. \$QK and \$QL use common logic.

Floating subtract

Identification: \$QL.

Computes difference of two floating point numbers.

Floating-point multiply or divide

Identification: \$QM, \$QN.

Multiplies 2 floating point numbers. Divides one number by another. Separates the mantissa and use XDMU for multiply or XDDI for divide. Uses \$FMS, \$SE.

Integer number to floating-point number

Identification: \$QS.

Floats an integer. Formats the absolute number to floating point and adjusts sign according to input. Uses \$SE.

Fixed single-precision logarithm

Identification: XLOG.

Computes the natural logarithm of $1 + X$, where the single-precision quantity X is in the A register. If

$$0 \leq X < 1,$$

the result is returned to the A register, otherwise an error exit is taken without further action. Input and output are scaled by 2^0 . Uses a Chebychev polynomial of the fifth degree.

Fixed single-precision exponential, positive arguments

Identification: XEXP.

Computes the exponential of X , located in the A register:

$$e^X, 0 \leq X < 1$$

e^X is scaled 2^{-2} . The result is placed in the A register. (Also see PURPOSE in subroutine XEXN.) The exponential is performed by means of a Chebychev poly-

nomial of the fifth degree.

Fixed single exponential, negative argument

Identification: XEXN.

Computes the exponential of X, located in the A register:

$$e^X, -1 < X < 0$$

e^X is scaled 2^0 . The result is placed in the A register. (Also see purpose in subroutine XEXP.) The exponential was split into two subroutines, XEXP and XEXN, to increase scaling flexibility. The exponential is performed by means of a Chebychev polynomial of the fifth degree.

Fixed single-precision square root (short)

Identification: XSQT.

Takes the unrounded square root of the quantity in the A register if it is non-negative. The result is returned to the A register. The A register is unchanged if the input is negative.

Fixed single-precision sine

Identification: XSIN.

Takes the sine of the quantity X in the A register for range $-\pi \leq X \leq \pi$. The input is scaled by 2^{-2} . The output is returned to the A register, scaled 2^{-1} . X is in radians. Uses a change of variable to y to reduce range from $(-\pi, \pi)$ to $(-\pi/2, \pi/2)$. The change of variable is $\sin x = \sin y$.

$$y = \left| X - \frac{\pi}{2} \right| - \frac{\pi}{2} \text{ if } X \geq 0$$

$$y = \left| X - \frac{\pi}{2} \right| + \frac{\pi}{2} \text{ if } X < 0$$

The Taylor sine series, truncated to five terms, is used for $\sin y$.

Fixed single-precision cosine

Identification: XCOS.

Takes the cosine of the quantity X in A register from range $-\pi \leq X \leq \pi$. The input is scaled by 2^{-2} and the output is scaled by 2^{-1} . The output is returned to the A register. Uses a change of variable to y in order to reduce the range of the variable from $(-\pi, +\pi)$ to $-\pi/2, +\pi/2$. Then $\cos x = \sin y$, where $y = \pi/2 - X$. The Taylor sine series, truncated to five terms, is used for $\sin y$.

Fixed single-precision arctangent

Identification: XATN.

Takes the arctangent of the quantity X in the A register, where $-1 < X < 1$. The input is scaled times 2^0 and the output is scaled times 2^0 . Uses a Chebychev polynomial of seven terms. This polynomial is adequate for an 18-bit configuration.

Single-precision polynomial

Identification: POLY.

A resident utility routine intended primarily to support the fixed-point single-precision mathematical subroutines requiring the evaluation of a polynomial in one variable of any finite degree. The polynomial is evaluated in Horner form.

Natural log of floating-point number.

Identification: ALOG.

Computes natural log of a floating-point number. Uses \$ER, \$QS, \$QK, \$QM, XDMU, XDAD, \$FMS, \$NML, XDDI, XDSU, \$SE routines.

Arctangent of a floating-point number

Identification: ATAN.

Computes arctangent of radians in floating point. Uses \$QM, \$QL, \$QN, \$QK, \$SE routines.

Cosine

Identification: COS.

Computes cosine of angle in floating-point radians. Computes sine of ($\pi/2 - A$). Uses SIN, \$QL, \$SE. Output in A, B registers.

Exponential

Identification: EXP.

Computes e^{**A} . A is floating-point number. Chebychev approximation uses XDMU, \$QK, \$QL, \$QM, \$QN, \$SE.

Sine

Identification: SIN.

Computes sine of radians in floating point. First 5 terms of Taylor series expansion output in A, B registers. Uses \$NML, \$QM, XDMU, XDAD, \$SE, \$FMS.

Square root

Identification: SQRT.

Computes square root of a floating point number. Newton iteration three times. Uses \$SE, XDDI, \$FMS.

Exponentiation of two integers

Identification: \$HE.

Computes I^{**J} , I and J are integers. Floats I and uses \$PE. Uses \$SE, \$QS, \$HS, \$PE.

Exponentiation

Identification: \$PE.

Computes A^{**I} , A is a floating-point number, I is an integer. Uses \$QS, \$QE, and \$SE. Floats I and goes to A^{**B} (\$QE).

Exponentiation

Identification: \$QE.

Computes A^{**B} , antilog of $B \log A = e^{** (B \log A)}$. Uses ALOG, EXP, \$SE.

Fixed single-precision integer binary-to-decimal conversion

Identification: XBTD.

Converts the absolute value of the integer in the A register, modulo 10,000, to a four digit decimal coded integer in the B register. The input is retained in the A register and the X register is unchanged. The output range is 0 through 9999 inclusive. Uses successive division of binary integer by 10_{10} with concatenation of remainders.

Fixed single-precision integer decimal-to-binary conversion

Identification: XDTB.

Converts the four-digit binary-coded decimal integer in the A register to a binary integer in the B register. The input is retained in the A register with the X register

unchanged. The input range is +0 through +9999 inclusive. Uses successive multiplication of digits by powers of 10 with accumulation.

$$B = ((10D_3 + D_2)_{10} \rightarrow 1) 10 + D_0.$$

Converts EBCDIC to Hollerith

Identification: SA01

Converts an EBCDIC character in bits 0 through 7 of the A register to 029 Hollerith code in bits 0 through 11 of the A register.

Convert Hollerith to EBCDIC

Identification: SB01

Converts an 029 Hollerith character in bits 0 through 11 of the A register to its corresponding EBCDIC code in bits 0 through 7 of the A register.

Converts EBCDIC to ASCII +

Identification: SC01

Converts an 8-bit EBCDIC character in bits 0 through 7 of the A register to its equivalent 8-bit ASCII code in bits 0 through 7 in the A register.

SECTION 26 – COMPUTER DIAGNOSTICS (MAINTAIN II)

MAINTAIN II provides the user with the ability to determine that hardware is functioning properly. The system is divided into preliminary and comprehensive tests of CPU, memory, options, and peripherals. Once assured that basic operations are working, the user can isolate malfunctions to specific CPU, memory, or peripheral equipment by using the appropriate comprehensive test program. MAINTAIN II is available in punched paper tape or card object decks.

The system is designed to operate with any hardware configuration. However, the minimum hardware configuration is a basic Varian 620/f computer with 4K of memory and a peripheral device to read the object program.

MAINTAIN II is detailed in the Varian MAINTAIN II Reference Manual (98 A 9908 960).

Test Executive Program

A composite test executive tape is structured into four logical parts:

- a. Preliminary instructions test
- b. Preliminary memory test
- c. Binary loader
- d. Test executive

The first two parts of the program perform preliminary checks of basic machine instructions and read/write and addressing to memory. If an error is encountered, the error condition is register-displayed.

After the preliminary tests are executed, the binary loader reads data from an object tape and stores the data in specified memory addresses. The program dynamically adapts to the device addresses used in the bootstrap loader. The binary loader program checks sums each record and compares the result with the value in the input record. If an error is detected, the program stops, allowing user intervention.

Communicating with the Test Executive

Once preliminary checks have been made and binary tape loading has been accomplished, the test executive is loaded. The test executive for MAINTAIN II provides test control and contains standard sub-routines that are common to subsequent MAINTAIN II test programs. Modular construction permits flexibility when future design improvements or additional tests are required for special systems.

The user communicates with the test executive through the TTY keyboard and printer. Statements and parameters input from the keyboard control execution and monitoring of associated test programs. To accommodate 4K memory systems, the test executive operates with only one test program in memory at a time.

Under user control, the executive loads test tapes, executes test programs, and provides a set of utility routines. The executive also provides the associated test programs with standardized routines, including TTY input/output.

Utility Routines

The test executive also provides software aids that are useful for debugging, program maintenance, and hardware troubleshooting. CPU registers and memory can be displayed

or altered. Programs can be trapped into or out of user-designated memory addresses. Data patterns can be searched for in memory, or specified patterns can be placed in memory. There are routines that also permit punching object paper tapes.

SECTION 27 – VORTEX

The VORTEX (Varian Omnitask Real-Time Executive) operating system is a powerful software package that significantly increases the usefulness and value of a Varian 620 computer installation. The system custom tailors computer hardware, rotating memories, and peripheral devices into a true Real-Time Multi-Programming system.

With VORTEX software controlling the system, the user is freed from such chores as I/O handling, interrupt servicing, and resource allocation. These operations are handled instead by a Real-Time Executive (RTE) task stored in the computer's main memory.

The RTE enables the user to assign priorities to real-time foreground tasks, which may be loaded and run either immediately or at some future time in response to the real-time clock, an external interrupt, the operator, or an inter-task command.

In addition, the RTE performs the house-keeping functions that permit general-purpose background tasks to be run concurrently with the real-time foreground operations in a multi-programming environment.

This multi-programming capability of the VORTEX system takes advantage of normally wasted periods of "idle time" which are embedded in most programs. Since such periods often constitute a large percentage of the run time of a program, VORTEX may actually double system hardware utilization by using these periods to run lower priority tasks such as background programs for compilation, debugging or general utility.

The fact that all foreground tasks and the VORTEX nucleus run in protected memory insures system integrity for the real-time application. At the same time, the background user has such advantages as easier software debug (e.g., his program may "blowup" in the background without disturbing system operation or necessitating lengthy "cold-start" and reload operations.)

The overall result of the VORTEX operating system will be to provide both real-time and background users with a more convenient and comprehensive control over system operations than has been available to any minicomputer user up to this time.

Features

True Multi-Task Capability — Multiple foreground tasks may execute concurrently with each other and with a background task, for maximum hardware utilization. The number of foreground tasks is limited only by available memory space.

Real-Time Foreground Operation — Foreground tasks are scheduled and run by the Real Time Executive in response to Real Time Clock, operator requests, external interrupts, or requests from other tasks.

Foreground/Background Overlay Capability — Large programs can be easily segmented and overlayed in both foreground and background areas, for more efficient utilization of main memory.

Automatic Background Scheduling — The Job Control Processor permits automatic background scheduling from the job stream, without operator intervention.

Simplified I/O Operations — Re-entrant I/O drivers, callable from any task by means of logical unit and file names, permit simple, hardware-independent I/O programming.

Protected Main Memory — Hardware memory protection for foreground and operating-system areas insures system integrity and allows for simple background program debugging.

Dynamic Memory Allocation — Efficient operation is insured by dynamic allocation of main memory area and by a moveable foreground/background boundary.

Comprehensive Operator Communications — The OpCom Component permits operator monitoring and control of job status, program priorities, peripheral status, and hardware interrupts at any time.

Tailored Software Configurations — The System Generator lets the user custom-tailor VORTEX to his particular application.

SECTION 28 — MASTER OPERATING SYSTEM (MOS)

The MOS is a batch-processing operating system for the Varian 620 computer. MOS operates on a wide range of hardware configurations. It is modular, thus facilitating expansion (e.g., new language processors, special user I/O drivers, etc.). MOS makes optimum use of memory by loading only those portions of the system (including I/O) required during execution. Features of MOS include:

- a. Minimum operator intervention
- b. Single tape, drum, or disc as secondary storage
- c. Extensive job control language (22 directives)
- d. Multisource input during loading
- e. Debugging aids
- f. Source file editing program
- g. System maintenance program

MOS requires, in addition to the computer, an 8K memory, a 33/35 ASR TTY, and either a rotating memory unit (on a buffer interlace controller) or a magnetic tape unit. MOS supports and is enhanced by the card reader, line printer, hardware multiply/divide, high-speed paper tape and/or card punch, and additional memory increments.

MOS is divided into resident and non-resident partitions. The resident partition comprises the resident monitor, absolute loader, I/O assignment tables, system flags

and parameters, and the dump.

The nonresident partition comprises the control programs, support programs, and language processors. The control programs are:

- a. Executive — job control processor and system control
- b. System loader — linking and relocating loading of programs
- c. I/O control — dispatching of I/O requests and device driving

The support programs are:

- a. Mathematical and support library
- b. Concordance program
- c. Debugging program
- d. Source file editing program
- e. System maintenance program
- f. System preparation program (operates in stand-alone mode)

The language processors are:

- a. DAS MR, macro assembler
- b. FORTRAN IV Compiler (requires an additional memory increment)

The executive program executes or directs

master operating system

the execution of some 22 system directives. It interprets and, executes directives serially either directly, or by calling and executing the appropriate software. Upon completion of each directive, the executive resumes control and goes to the next directive.

The I/O control program processes all I/O requests. I/O control is flexible and device-independent. It assigns logical units (e.g., system file, system input, list output, etc.) to peripherals instead of addressing the devices as hardware units. Up to 255 logical units can be assigned. The units are linked to peripherals by user directives and device assignment tables. Assignments can be changed at any time. Thus, different peripherals can be substituted for I/O operations without reassembling the program. The same peripheral can serve as more than one logical unit, and multiple devices of a given type can be used. I/O for logical devices is serviced by I/O drivers. Only those

drivers required for a given program are in memory during execution.

Under MOS supervision, the DAS MR macro assembler and FORTRAN IV compiler automatically process source statements to generate relocatable and MOS compatible object programs. These can be checked for errors with the MOS debugging program, which includes directives for memory searching, interrogation or alteration of logical units, core dump, etc.

The system maintenance program edits the MOS library. Programs can be added to or deleted from the library. The source file editing program permits creation, duplication, and correction of source files, such as symbolic DAS and FORTRAN IV program statements.

MOS is detailed in the Master Operating System Reference Manual (98 A 9952 090).

SECTION 29 – FORTRAN IV

Varian FORTRAN IV is a programming system that permits simple solutions of complex mathematical problems. It permits users having little computer experience to effectively utilize the computational power of the Varian 620/f. Problems can be stated in simple English words and mathematical terms. Thus, the learning period required for useful programming is short.

The FORTRAN IV language is fully compatible with the American National Standards Institute (ANSI) FORTRAN.

FORTRAN IV is available as a stand-alone system or as a component of MOS and VORTEX. The principal components of the system are a compiler, a loader, and a comprehensive run-time package. The one-pass processing provides convenient, efficient compilations and the entire system operates in only 8K of memory. The MOS and VORTEX systems requires 12K. The run-time module contains input/output driver routines for the full line of Varian 620 peripherals, plus arithmetic and mathematical function libraries.

The principal features of the FORTRAN IV language include:

- a. Implied DO loop in DATA initialization statement

- b. BLOCK DATA statement
- c. Array declarators permitted in COMMON statements
- d. Labelled COMMON specification
- e. Adjustable DIMENSION capability
- f. Three-dimensional arrays and subscripts permitted
- g. Double-precision and complex arithmetic
- h. Relational operators, .LT., .LE., .GT., .GE., .EQ., .NE.
- i. Explicit function and data identifiers, INTEGER, REAL, DOUBLE PRECISION, COMPLEX, and LOGICAL
- k. A, F, E, D, I, H, X, L, and G field specifications in a FORMAT statement
- l. EXTERNAL statement
- m. Single- or double-word integers and logical data

FORTRAN IV is detailed in the FORTRAN IV Reference Manual (98 A 9902 037).

SECTION 30 – BASIC

BASIC is a popular, easy-to-use programming language for a wide variety of business and scientific applications. The simplicity of BASIC, plus its conversational operation, permit the inexperienced operator to perform useful programming with just a few hours of training. The computer responds to all commands entered by the user, thus reinforcing the learning process. If the operator makes an error, diagnostics report the type of error, and corrections can be made immediately.

For the experienced programmer, the Varian version of BASIC includes expanded instructions and capabilities. The advanced features permit wider-range programming applications, retaining the inherent simplicity of the language.

Only 8K memory and a TTY are required for using BASIC in Varian 620/f system. Even dedicated computers can perform general computation when not used for other primary functions.

The distinguishing features of BASIC include:

- a. Arithmetic operators +, -, *, /, ↑
- b. Logical operators AND, OR, NOT
- c. Relational operators >, <, ≥, ≤, =, ≠

- d. Optional computed GOTO and GOSUB statements
- e. Parameter-passing on GOSUB allowed
- f. SUB statement for defining parameters
- g. CALL statement for linking to assembler language subroutines
- h. Recursive execution of subroutines allowed
- i. STOP statement causes a program pause
- j. Complete matrix operations and trig functions

Operational features of the language are:

- a. Immediate syntax-checking and diagnosis of statements
- b. Immediate, single-statement execution when requested (calculator mode)
- c. Optional TTY or high speed paper tape I/O
- d. Optimum memory utilization through operator library selection

BASIC is detailed in the Varian 620 BASIC Reference Manual (98 A 9952 031).

SECTION 31 — REPORT PROGRAM GENERATOR IV (RPG IV)

RPG IV is a business-oriented language and system for the 620/f computer system. It simply and efficiently produces problem-solving and report-generation programs.

The RPG IV compiler is an advanced version of RPG systems now widely used throughout industry for commercial applications. Application programs written in the RPG IV language are far more concise than the equivalent programs written in COBOL. In typical instances, only a fourth or a third of the program steps are required, resulting in faster processing and a reduction in the amount of memory required to store the program.

RPG IV improves on common RPG languages by providing many automatic features as well as powerful procedural statements. Each statement is written free-form, simplifying the programmer's task.

An application program written in the RPG IV language consists of a series of statements. Each statement is a concise directive; the number of names and labels is small, yet meaningful. Such functions as automatic scaling of numeric data, sequence-checking, control break testing, and auditing of input records and fields are built into the language, relieving the programmer of these tasks through multiple directives.

Two-Stage Operation

The Varian RPG IV system operates in two stages.

First Stage

The first stage is the preparation of the

specific application program that will produce a required report or an update of an existing report. To use RPG IV, a series of statements are written, defining the data to be processed, calculations to be performed, and the contents and format of the finished report or output file. These statements, which make up the RPG IV source program, are transferred to punched cards, one statement per card.

Using a loader program supplied by Varian, the RPG IV compiler program is loaded into the computer memory. The cards representing the RPG IV source program are then read by the card reader and the individual statements processed by the compiler program to produce a second deck of punched cards representing the RPG IV object program.

The object program is a translation of the source program. It consists of a series of machine instructions that are executed by the computer in performing the application. The computer also produces a printed output of the original source program for reference and debugging.

Second Stage

The second stage is the loading and execution of the RPG IV object program that does the processing of actual data into finished reports or output files. The object program produced by the first stage is read into memory, along with a run-time support program supplied by Varian. The users' data files on punched cards are then read by the card reader and processed by the computer.

RPG IV

Printed Output

The principal output is a printed record, prepared on the line printer, in finished form ready for distribution.

In certain applications, the program can also output a portion or all of the processed data, on punched cards, to be used as input

data for later processing. Typical would be a record of an updated end-of-the-month inventory, which would then be used as a start-of-the-month inventory record at the end of the following month.

RPG IV is detailed in the Varian RPG IV system user's manual (98A 9947 030).

APPENDIX

Contents	Page
Standard Character Codes	A-2
TTY Character Codes	A-5
Powers of Two	A-7
Octal-Decimal Integer Conversions	A-8
Octal-Decimal Fraction Conversions	A-12
Alphabetical Index of Instructions	A-15
Numerical Index of Instructions	A-23

standard character codes

Symbol	ASCII	Printer*	Mag Tape	Hollerith	FORTTRAN
@	300	100	32	0-2-8	77
A	301	101	61	12-1	13
B	302	102	62	12-2	14
C	303	103	63	12-3	15
D	304	104	64	12-4	16
E	305	105	65	12-5	17
F	306	106	66	12-6	20
G	307	107	67	12-7	21
H	310	110	70	12-8	22
I	311	111	71	12-9	23
J	312	112	41	11-1	24
K	313	113	42	11-2	25
L	314	114	43	11-3	26
M	315	115	44	11-4	27
N	316	116	45	22-5	30
O	317	117	46	11-6	31
P	320	120	47	11-7	32
Q	321	121	50	11-8	33
R	322	122	51	11-9	34
S	323	123	22	0-2	35
T	324	124	23	0-3	36
U	325	125	24	0-4	37
V	326	126	25	0-5	40
W	327	127	26	0-6	41

*Varian 620-77

Standard Character Codes (1 of 3)

Symbol	ASCII	Printer*	Mag Tape	Hollerith	FORTRAN
X	330	130	27	0-7	42
Y	331	131	30	0-8	43
Z	332	132	31	0-9	44
[	333	133	75	12-5-8	76 ¹
\	334	134	36	0-6-8	76 ¹
]	335	135	55	11-5-8	76 ¹
↑	336	136	17	7-8	76 ¹
			(Note)		
←	337	137	20	2-8	76 ²
blank	240	40	20	No Punch	00
!	241	40	52	11-2-8	51
''	242	42	35	0-5-8	62
#	243	43	37	0-7-8	63
\$	244	44	53	11-3-8	60
%	245	45	57	11-7-8	64
&	246	46	77	12-7-8	65
'	247	47	14	4-8	66
(	250	50	34	0-4-8	52
)	251	51	74	12-4-8	53
*	252	52	54	11-4-8	47
+	253	53	60	12	45
,	254	54	33	0-3-8	54
-	255	55	40	11	46
.	256	56	73	12-3-8	51
/	257	57	21	0-1	50

Standard Character Codes (2 of 3)

standard character codes

Symbol	ASCII	Printer*	Mag Tape	Hollerith	FORTTRAN
0	260	60	12	0	01
1	261	61	01	1	02
2	262	62	02	2	03
3	263	63	03	3	04
4	264	64	04	4	05
5	265	65	05	5	06
6	266	66	06	6	07
7	267	67	07	7	10
8	270	70	10	8	11
9	271	71	11	9	12
:	272	72	15	5-8	67
;	273	73	56	11-6-8	70
<	274	74	76	12-6-8	76 ¹
=	275	75	13	3-8	55
>	276	76	16	6-8	76 ³
?	277	77	72	12-2-8	76
Note: End-of-file for mag tape. 1: Undefined character. 2: Form control: Return to col 1. 3: Tab control: Skip to col 7.					
FORTTRAN System only					

Standard Character Codes (3 of 3)

Teletype Character	Varian 620/f Internal Code	Teletype Character	Varian 620/f Internal Code
0	260	Y	331
1	261	Z	332
2	262	blank	240
3	263	!	241
4	264	'	242
5	265	#	243
6	266	\$	244
7	267	%	245
8	270	&	246
9	271	'	247
A	301	(	250
B	302	)	251
C	303	*	252
D	304	+	253
E	305	,	254
F	306	.	255
G	307	-	256
H	310	/	257
I	311	:	272
J	312	;	273
K	313		274
L	314	=	275
M	315		276
N	316	?	277
O	317	@	300
P	320		333
Q	321		334
R	322		335
S	323		336
T	324		337
U	325	Rub Out	377
V	326	NUL	200
W	327	SOM	201
X	330	EOA	202

Teletypewriter Character Codes (1 of 2)

TTY character codes

Teletype Character	Varian 620/f Internal Code	Teletype Character	Varian 620/f Internal Code
EOM	203	X-OFF	223
EOT	204	TAPE OFF	
WRU	205	AUX	224
RU	206	ERROR	225
BEL	207	SYNC	226
FE	210	LEM	227
H TAB	211	SO	230
LINE FEED	212	S1	231
V TAB	213	S2	232
FORM	214	S3	233
RETURN	215	S4	234
SO	216	S5	235
SI	217	S6	236
DCO	220	S7	237
X-ON	221		
TAPE AUX			
ON	222		

Teletypewriter Character Codes (2 of 2)

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.015625
128	7	0.0078125
256	8	0.00390625
512	9	0.001953125
1024	10	0.0009765625
2048	11	0.00048828125
4096	12	0.000244140625
8192	13	0.0001220703125
16384	14	0.00006103515625
32768	15	0.000030517578125
65536	16	0.0000152587890625
131072	17	0.00000762939453125
262144	18	0.000003814697265625
524288	19	0.0000019073486328125
1048576	20	0.00000095367431640625
2097152	21	0.000000476837158203125
4194304	22	0.0000002384185791015625
8388608	23	0.00000011920928955078125
16777216	24	0.000000059604644775390625
33554432	25	0.0000000298023223876953125
67108864	26	0.00000001490116119384765625
134217728	27	0.000000007450580596923828125
268435456	28	0.0000000037252902984619140625
536870912	29	0.00000000186264514923095703125
1073741824	30	0.000000000931322574615478515625
2147483648	31	0.0000000004656612873077392578125
4294967296	32	0.00000000023283064365386962890625
8589934592	33	0.000000000116415321826934814453125
17179869184	34	0.0000000000582076609134674072265625
34359738368	35	0.00000000002910383045673370361328125
68719476736	36	0.000000000014551915228366851806640625
137438953472	37	0.0000000000072759576141834259033203125
274877906944	38	0.00000000000363797880709171295166015625
549755813888	39	0.000000000001818989403545856475830078125

Powers of Two

octal decimal conversions

0000 to 0777 (Octal)
0000 to 0511 (Decimal)
10000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

1000 to 1777 (Octal)
0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

octal decimal conversions

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087

2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151

2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215

2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599

3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663

3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727

3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343

2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407

2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471

2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855

3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919

3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983

3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

2000 to 2777 (Octal)
1024 to 1535 (Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

3000 to 3777 (Octal)
1536 to 2047 (Decimal)

	0	1	2	3	4	5	6	7
4000 to 4777 (Octal)	2048 to 7359 (Decimal)							
Octal	Decimal							

10000 - 4096								
20000 - 8192								
30000 - 12288								
40000 - 16384								
50000 - 20480								
60000 - 24576								
70000 - 28672								

4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111

4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175

4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239

4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367

4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431

4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495

4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

	0	1	2	3	4	5	6	7
5000 to 5777 (Octal)	2560 to 3071 (Decimal)							
Octal	Decimal							

5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623

5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687

5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751

5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879

5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943

5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007

5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135

6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199

6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263

6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647

7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711

7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775

7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391

6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455

6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519

6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903

7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967

7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031

7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

6000 to 6777 (Octal) 3072 to 3583 (Decimal)

Octal Decimal
10000 - 4096
20000 - 8192
30000 - 12288
40000 - 16384
50000 - 20480
60000 - 24576
70000 - 28672

7000 to 7777 (Octal) 3584 to 4095 (Decimal)

octal decimal conversions

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Fraction Conversions (1 of 3)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000206	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Fraction Conversions (2 of 3)

octal decimal conversions

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

Octal-Decimal Fraction Conversions (3 of 3)

Mnemonic	Octal	Definition	Page
ADD	120000	Add to A Register	20-7
ADDE	00612Z	Add to A Register Extended	20-7
ADDI	006120	Add to A Register Immediate	20-7
ANA	150000	AND to A Register	20-14
ANAE	00615Z	AND to A Register Extended	20-14
ANAI	006150	AND to A Register Immediate	20-14
AOFA	005511	Add OF to A Register	20-37
AOFB	005522	Add OF to B Register	20-37
AOFX	005544	Add OF to X Register	20-37
ASLA	004200+n	Arithmetic Shift Left A n Places	20-31
ASLB	004000+n	Arithmetic Shift Left B n Places	20-32
ASRA	004300+n	Arithmetic Shift Right A n Places	20-31
ASRB	004100+n	Arithmetic Shift Right B n Places	20-32
BT	006400	Jump Unconditionally and set return in Index Register	20-18d
CIA	1025XX	Clear and Input to A Register	20-40
CIAB	1027XX	Clear and Input to A and B	20-41
CIB	1026XX	Clear and Input to B Register	20-41
COMPL	005200	Complement Source to Destination	20-38
CPA	005211	Complement A Register	20-35
CPB	005222	Complement B Register	20-35

X = 0-7

Z = 4-7

Varian 620/f Instructions, Alphabetical Order (1 of 8)

Mnemonic	Octal	Definition	Page
CPX	005244	Complement X Register	20-35
DAR	005311	Decrement A Register	20-34
DBR	005322	Decrement B Register	20-34
DECR	005300	Decrement Source to Destination	20-38
DIV	170000	Divide AB Register	20-10
DIVE	00617Z	Divide AB Register Extended	20-10
DIVI	006170	Divide AB Register Immediate	20-10
DXR	005344	Decrement X Register	20-34
ERA	130000	Exclusive OR to A Register	20-13
ERAE	00613Z	Exclusive OR to A Register Extended	20-13
ERAI	006130	Exclusive OR to A Register Immediate	20-13
EXC	100XXX	External Control Function	20-40
HLT	000XXX	Halt	20-28
IAR	005111	Increment A Register	20-34
IBR	005122	Increment B Register	20-34
IJMP	006700	Indexed Jump	20-18b
IME	1020XX	Input to Memory	20-43
INA	1021XX	Input to A Register	20-41
INAB	1023XX	Input to A and B Registers	20-42
INB	1022XX	Input to B Register	20-41
INCR	005100	Increment Source to Destination	20-38

Mnemonic	Octal	Definition	Page
INR	040000	Increment and Replace	20-6
INRE	00604Z	Increment and Replace Extended	20-6
INRI	006040	Increment and Replace Immediate	20-6
IXR	005144	Increment X Register	20-34
JAN	001004	Jump if A Register Negative	20-16
JANM	002004	Jump and Mark if A Register Negative	20-20
JANZ	001016	Jump if A Register Register not zero	20-18a
JANZM	002016	Jump and Mark if A Register not zero	20-22a
JAP	001002	Jump if A Register Positive	20-16
JAPM	002002	Jump and Mark if A Register Positive	20-20
JAZ	001010	Jump if A Register Zero	20-17
JAZM	002010	Jump and Mark if A Register Zero	20-21
JBNZ	001026	Jump if B Register Register not zero	20-18a
JBNZM	002026	Jump and Mark if B Register not zero	20-22a
JBZ	001020	Jump if B Register Zero	20-17
JBZM	002020	Jump and Mark if B Register Zero	20-21
JIF	001000	Jump if Combined Conditions	20-18
JIFM	002000	Jump and Mark if Combined Conditions	20-22
JMP	001000	Jump Unconditionally	20-16
JMPM	002000	Jump and Mark Unconditionally	20-20

instructions (alphabetical)

Mnemonic	Octal	Definition	Page
JOF	001001	Jump if Overflow On	20-16
JOFN	001007	Jump if Overflow not set	20-18c
JOFM	002001	Jump and Mark if Overflow On	20-20
JOFNM	002007	Jump and Mark if Overflow not set	20-22a
JSR	006500	Jump Unconditionally and set return in Index Register	20-18c
JS1M	002100	Jump and Mark if Sense Switch 1 On	20-21
JS2M	002200	Jump and Mark if Sense Switch 2 On	20-22
JS3M	002400	Jump and Mark if Sense Switch 3 On	20-22
JS1N	001106	Jump if Sense Switch 1 not set	20-18b
JS2N	001206	Jump if Sense Switch 2 not set	20-18b
JS3N	001406	Jump if Sense Switch 3 not set	20-18b
JS1NM	002106	Jump and Mark if if Switch 1 not set	20-22b
JS2NM	002206	Jump and Mark if Sense Switch 2 not set	20-22b
JS3NM	002406	Jump and Mark if Sense Switch 3 not set	20-22b
JSS1	001100	Jump if Sense Switch 1 On	20-17
JSS2	001200	Jump if Sense Switch 2 On	20-18
JSS3	001400	Jump if Sense Switch 3 On	20-18
JXNZ	001046	Jump if X Register not zero	20-18a
JXNZM	002046	Jump and Mark if X Register not zero	20-22a
JXZ	001040	Jump X Register Zero	20-17
JXZM	002040	Jump and Mark X Zero	20-21

Varian 620/f Instructions, Alphabetical Order (4 of 8)

Mnemonic	Octal	Definition	Page
LASL	004400+n	Long Arithmetic Shift Left n Places	20-32
LASR	004500+n	Long Arithmetic Shift Right n Places	20-32
LDA	010000	Load A Register	20-2
LDAE	00601Z	Load A Register Extended	20-2
LDAI	006010	Load A Register Immediate	20-2
LDB	020000	Load B Register	20-2
LDBE	00602Z	Load B Register Extended	20-2
LDBI	006020	Load B Register Immediate	20-2
LDX	030000	Load X Register	20-3
LDXE	00603Z	Load X Register Extended	20-3
LDXI	006030	Load X Register Immediate	20-3
LLRL	004440+n	Long Logical Rotate Left n Places	20-31
LLSR	004540+n	Long Logical Shift Right n Places	20-31
LRLA	004240+n	Logical Rotate Left A n Places	20-30
LRLB	004040+n	Logical Rotate Left B n Places	20-30
LSRA	004340+n	Logical Shift Right A n Places	20-30
LSRB	004140+n	Logical Shift Right B n Places	20-30
MERGE	005000	Merge Source to Destination	20-38
MUL	160000	Multiply B Register	20-9
MULE	00616Z	Multiply B Register Extended	20-9
MULI	006160	Multiply B Register Immediate	20-9
NOP	005000	No Operation	20-28

OAB	1033XX	Output OR of A and B Registers	20-42
OAR	1031XX	Output from A Register	20-42
OBR	1032XX	Output from B Register	20-42
OME	1030XX	Output from Memory	20-43
ORA	110000	Inclusive OR to A Register	20-12
ORAE	00611Z	Inclusive OR to A Register Extended	20-12
ORAI	006110	Inclusive OR to A Register Immediate	20-12
ROF	007400	Reset Overflow	20-28
SEN	101XXX	Sense Input/Output Lines	20-40
SOF	007401	Set Overflow	20-28
SOFA	005711	Subtract OFLO from A Register	20-37
SOFB	005722	Subtract OFLO from B Register	20-37
SOFX	005744	Subtract OFLO from X Register	20-37
SRE	006600	Skip if Register Equal (optional)	20-18d
STA	050000	Store A Register	20-3
STAE	00605Z	Store A Register Extended	20-3
STAI	006050	Store A Register Immediate	20-3
STB	060000	Store B Register	20-4
STBE	00606Z	Store B Register Extended	20-4
STBI	006060	Store B Register Immediate	20-4
STX	070000	Store X Register	20-4
STXE	00607Z	Store X Register Extended	20-4

Mnemonic	Octal	Definition	Page
STXI	006070	Store X Register Immediate	20-4
SUB	140000	Subtract from A Register	20-8
SUBE	006140	Subtract from A Register Extended	20-8
SUBI	006140	Subtract from A Register Immediate	20-8
TAB	005012	Transfer A to B Register	20-35
TAX	005014	Transfer A to X Register	20-35
TBA	005021	Transfer B to A Register	20-35
TBX	005024	Transfer B to X Register	20-35
TSA	007402	Transfer Switches to A Register	20-5
TXA	005041	Transfer X to A Register	20-36
TXB	005042	Transfer X to B Register	20-36
TZA	005001	Transfer Zero to A Register	20-36
TZB	005002	Transfer Zero to B Register	20-36
TZX	005004	Transfer Zero to X Register	20-36
XAN	003004	Execute A Register Negative	20-24
XANZ	003016	Execute if A Register not zero	26a
XAP	003002	Execute A Register Positive	20-24
XAZ	003010	Execute A Register Zero	20-25
XBNZ	003026	Execute if B Register not zero	26a
XBZ	003020	Execute B Register Zero	20-25
XEC	003000	Execute Unconditionally	20-24

instructions (alphabetical)

Mnemonic	Octal	Definition	Page
XIF	003000	Execute if Combined Conditions	20-26
XOF	003001	Execute Overflow Set	20-24
XOFN	003007	Execute if Overflow not set	20-26
XS1	003100	Execute SENSE Switch 1 Set	20-25
XS2	003200	Execute SENSE Switch 2 Set	20-26
XS3	003400	Execute SENSE Switch 3 Set	20-26
XS1N	003106	Execute if Sense Switch 1 not set	26a
XS2N	003206	Execute if Sense Switch 2 not set	26b
XS3N	003406	Execute if Sense Switch 3 not set	26b
XXNZ	003046	Execute if X Register not zero	26a
XXZ	003040	Execute X Register Zero	20-25
ZERO	005000	Zero Register	20-39

Varian 620/f Instructions, Alphabetical Order (8 of 8)

Octal	Mnemonic	Definition	Page
000000	HLT	Halt	20-28
001000	JIF	Jump if Combined Conditions	20-18
001000	JMP	Jump Unconditionally	20-16
001001	JOF	Jump if Overflow On	20-16
001002	JAP	Jump if A Register Positive	20-16
001004	JAN	Jump if A Register Negative	20-16
001007	JOFN	Jump if Overflow not set	20-18c
001010	JAZ	Jump if A Register Zero	20-17
001016	JANZ	Jump if A Register not zero	20-18a
001020	JBZ	Jump if B Register Zero	20-17
001026	JBNZ	Jump in B Register not zero	20-18a
001040	JXZ	Jump X Register Zero	20-17
001046	JXNZ	Jump if X Register not zero	20-18A
001100	JSS1	Jump if Sense Switch 1 On	20-17
001106	JS1N	Jump if Sense Switch 1 not set	20-18b
001200	JSS2	Jump if Sense Switch 2 On	20-18
001206	JS2N	Jump if Sense Switch 2 not set	20-18b
001400	JSS3	Jump if Sense Switch 3 On	20-18
001406	JS3N	Jump if Sense Switch 3 not set	20-18b
002000	JIFM	Jump and Mark if Combined Conditions	20-22
002000	JMPM	Jump and Mark Unconditionally	20-20

instructions (numerical)

Octal	Mnemonic	Defintion	Page
002001	JOFM	Jump and Mark if Overflow On	20-20
002002	JAPM	Jump and Mark if A Register Positive	20-20
002004	JANM	Jump and Mark if A Register Negative	20-20
002007	JOFNM	Jump and Mark if Overflow not set	20-22a
002010	JAZM	Jump and Mark if A Register Zero	20-21
002016	JANZM	Jump and Mark if A Register not zero	20-22a
002020	JBZM	Jump and Mark if B Register Zero	20-21
002026	JBNZM	Jump and Mark if B Register not zero	20-22a
002040	JXZM	Jump and Mark X Zero	20-21
002046	JXNZM	Jump and Mark if X Register not zero	20-22a
002100	JS1M	Jump and Mark if Sense Switch 1 On	20-21
002106	JS1NM	Jump and Mark if Sense Switch 1 not set	20-22b
002200	JS2M	Jump and Mark if Sense Switch 2 On	20-22
002206	JS2NM	Jump and Mark if Sense Switch 2 not set	20-22b
002400	JS3M	Jump and Mark if Sense Switch 3 On	20-22
002406	JS3NM	Jump and Mark if Sense Switch 3 not set	20-22b
003000	XEC	Execute Unconditionally	20-24
003000	XIF	Execute if Combined Conditions	20-26
003001	XOF	Execute Overflow Set	20-24
003002	XAP	Execute A Register Positive	20-24

Varian 620/f Instructions, Numerical Order (2 of 8)

Octal	Mnemonic	Definition	Page
003004	XAN	Execute A Register Negative	20-24
003007	XOFN	Execute if Overflow not set	20-26
003010	XAZ	Execute A Register Zero	20-25
003016	XANZ	Execute if A Register not zero	26a
003020	XBZ	Execute B Register Zero	20-25
003026	XBNZ	Execute if B Register not zero	26a
003040	XXZ	Execute X Register Zero	20-25
003046	XXNZ	Execute if X Register not zero	26a
003100	XS1	Execute SENSE Switch 1 Set	20-25
003106	XS1N	Execute if Sense Switch 1 not set	26a
003200	XS2	Execute SENSE Switch 2 Set	20-26
003206	XS2N	Execute if Sense Switch 2 not set	26b
003400	XS3	Execute SENSE Switch 3 Set	20-26
003406	XS3N	Execute if Sense Switch 3 not set	26b
004000+n	ASLB	Arithmetic Shift Left B n Places	20-32
004040+n	LRLB	Logical Rotate Left B n Places	20-30
004100+n	ASRB	Arithmetic Shift Right B n Places	20-32
004140+n	LSRB	Logical Shift Right B n Places	20-30
004200+n	ASLA	Arithmetic Shift Left A n Places	20-31

instructions (numerical)

Octal	Mnemonic	Definition	Page
004240+n	LRLA	Logical Rotate Left A n Places	20-30
004300+n	ASRA	Arithmetic Shift Right A n Places	20-31
004340+n	LSRA	Logical Shift Right A n Places	20-30
004400+n	LASL	Long Arithmetic Shift Left n Places	20-32
004440+n	LLRL	Long Logical Rotate Left n Places	20-31
004500+n	LASR	Long Arithmetic Shift Right n Places	20-32
004540+n	LLSR	Long Logical Shift Right n Places	20-31
005200	COMP	Complement Source to Destination	20-38
005300	DECR	Decrement Source to Destination	20-38
005100	INCR	Increment Source to Destination	20-38
005000	MERG	Merge Source to Destination	20-38
005000	NOP	No Operation	20-28
005000	ZERO	Zero Register	20-39
005001	TZA	Transfer Zero to A Register	20-36
005002	TZB	Transfer Zero to B Register	20-36
005004	TZX	Transfer Zero to X Register	20-36
005012	TAB	Transfer A to B Register	20-35
005014	TAX	Transfer A to X Register	20-35
005021	TBA	Transfer B to A Register	20-35
005024	TBX	Transfer B to X Register	20-35

Varian 620/f Instructions, Numerical Order (4 of 8)

Octal	Mnemonic	Definition	Page
005041	TXA	Transfer X to A Register	20-36
005042	TXB	Transfer X to B Register	20-36
005111	IAR	Increment A Register	20-34
005122	IBR	Increment B Register	20-34
005144	IXR	Increment X Register	20-34
005211	CPA	Complement A Register	20-35
005222	CPB	Complement B Register	20-35
005244	CPX	Complement X Register	20-35
005311	DAR	Decrement A Register	20-34
005322	DBR	Decrement B Register	20-34
005344	DXR	Decrement X Register	20-34
005511	AOFA	Add OF to A Register	20-37
005522	AOFB	Add OF to B Register	20-37
005544	AOFX	Add OF to X Register	20-37
005711	SOFA	Subtract OFLO from A Register	20-37
005722	SOFB	Subtract OFLO from B Register	20-37
005744	SOFX	Subtract OFLO from X Register	20-37
00601Z	LDAE	Load A Register Extended	20-2
006010	LDAI	Load A Register Immediate	20-2
00602Z	LDBE	Load B Register Extended	20-2
006020	LDBI	Load B Register Immediate	20-2

instructions (numerical)

Octal	Mnemonic	Definition	Page
00603Z	LDXE	Load X Register Extended	20-3
006030	LDXI	Load X Register Immediate	20-3
00604Z	INRE	Increment and Replace Extended	20-6
006040	INRI	Increment and Replace Immediate	20-6
00605Z	STAE	Store A Register Extended	20-3
006050	STAI	Store A Register Immediate	20-3
00606Z	STBE	Store B Register Extended	20-4
006060	STBI	Store B Register Immediate	20-4
00607Z	STXE	Store X Register Extended	20-4
006070	STXI	Store X Register Immediate	20-4
00611Z	ORAE	Inclusive OR to A Register Extended	20-12
006110	ORAI	Inclusive OR to A Register Immediate	20-12
00612Z	ADDE	Add to A Register Extended	20-7
006120	ADDI	Add to A Register Immediate	20-7
00613Z	ERAE	Exclusive OR to A Register Extended	20-13
006130	ERAI	Exclusive OR to A Register Immediate	20-13
00614Z	SUBE	Subtract from A Register Extended	20-8
006140	SUBI	Subtract from A Register Immediate	20-8
00615Z	ANAE	AND to A Register Extended	20-14
006150	ANAI	AND to A Register Immediate	20-14

Octal	Mnemonic	Definition	Page
00616Z	MULE	Multiply B Register Extended	20-9
006160	MULI	Multiply B Register Immediate	20-9
00617Z	DIVE	Divide AB Register Extended	20-10
006170	DIVI	Divide AB Register Immediate	20-10
006400	BT	Jump Unconditionally and set return in Index Register	20-18d
006500	JSR	Jump Unconditionally and set return in Index Register	20-18c
006600	SRE	Skip if Register Equal (optional)	20-18d
006700	IJMP	Indexed Jump	20-18b
007400	ROF	Reset Overflow	20-28
007401	SOF	Set Overflow	20-28
007402	TSA	Transfer Switches to A Register	20-5
010000	LDA	Load A Register	20-2
020000	LDB	Load B Register	20-2
030000	LDX	Load X Register	20-3
040000	INR	Increment and Replace	20-6
050000	STA	Store A Register	20-3
060000	STB	Store B Register	20-4
070000	STX	Store X Register	20-4
100000	EXC	External Control Function	20-40
101000	SEN	Sense Input/Output Lines	20-40

instructions (numerical)

Octal	Mnemonic	Definition	Page
102000	IME	Input to Memory	20-43
102100	INA	Input to A Register	20-41
102200	INB	Input to B Register	20-41
102300	INAB	Input to A and B Registers	20-42
102500	CIA	Clear and Input to A Register	20-40
102600	CIB	Clear and Input to B Register	20-41
102700	CIAB	Clear and Input to A and B	20-41
103000	OME	Output from Memory	20-43
103100	OAR	Output from A Register	20-42
103200	OBR	Output from B Register	20-42
103300	OAB	Output OR of A and B Registers	20-42
110000	ORA	Inclusive OR to A Register	20-12
120000	ADD	Add to A Register	20-7
130000	ERA	Exclusive OR to A Register	20-13
140000	SUB	Subtract from A Register	20-8
150000	ANA	AND to A Register	20-14
160000	MUL	Multiply B Register	20-9
170000	DIV	Divide AB Register	20-10

Varian 620/f Instructions, Numerical Order (8 of 8)

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

(please detach card here)

First Class
Permit No. 323
Newport Beach
California

BUSINESS REPLY MAIL No Postage Stamp Necessary if Mailed in United States

postage will be paid by



varian data machines
2722 michelson drive
irvine/california/92664

varian data machines

2722 michelson drive
irvine/california/92664

INTERESTED IN:

- ☐ **Varian 620/f Computer**
- ☐ **Varian 620/L Computer**
- ☐ **Varian R 620/i Ruggedized Computer**
- ☐ **Varian 620/DC Data Communication System**

- ☐ Batch processing
- ☐ Analytical instrumentation data
- ☐ Telemetry or aerospace test data
- ☐ Small computer used as a local satellite to a larger computer for control of I/O peripherals
- ☐ Small computer used as a remote satellite to a larger computer for control of I/O peripherals
- ☐ Numerical control of machine tools or drafting machines
- ☐ Control of chemical, petroleum, or power generation processes
- ☐ Communication switching, data concentration, etc.
- ☐ Production line testing
- ☐ Other _____

(PLEASE SPECIFY)

- ☐ **Please have a sales engineer call me**
- ☐ **Please add my name to your mailing list**

Mail to:

NAME _____

TITLE _____

COMPANY _____ DEPT. CODE _____

ADDRESS _____

CITY _____ PHONE _____

STATE _____ ZIP _____

For VDM Use Only

\$3.00

varian **620/f** **computer** **handbook**



varian data machines

2722 Michelson Drive
Irvine, California 92664